HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Jiaqi Peng

Entwicklung und Aufbau eines Teleoperations-Demonstrators bestehend aus einem Roboter und einem Tablet-PC

Mittweida, 2018

Fakultät Ingenieurwissenschaften

BACHELORARBEIT

Entwicklung und Aufbau eines Teleoperations-Demonstrators bestehend aus einem Roboter und einem Tablet-PC

Autor:
Herr

Jiaqi Peng

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI16wA-BC

Erstprüfer:
Prof. Dr.-Ing. Alexander Winkler

Zweitprüfer:
M.Sc. Christian Thormann

Einreichung:
Mittweida, 15. August 2018

Verteidigung/Bewertung:
Mittweida, 2018

Faculty of engineering science

BACHELOR THESIS

Development and construction of a teleoperation demonstrator consisting of a robot and a tablet-PC

author:
Mr.

Jiaqi Peng

course of studies:
Electrical and information technology

seminar group:
EI16wA-BC

first examiner:
Prof. Dr.-Ing. Alexander Winkler

second examiner:
M.Sc. Christian Thormann

submission:
Mittweida, 15. August 2018

defence/ evaluation:
Mittweida, 2018

Bibliografische Beschreibung:

Peng, Jiaqi:
Entwicklung und Aufbau eines Teleoperations-Demonstrators bestehend aus einem Roboter und einem Tablet-PC. 08/2018, VII, 60, XXIV S.
Mittweida, Hochschule Mittweida, Fakultät Ingenieurwissenschaften, Bachelorarbeit, 2018

Abstrakt:

In der aktuellen industriellen Entwicklung werden neue Bedienkonzepte für die Robotik immer wichtiger. Tablet-PC's bieten hierfür geeignete Voraussetzungen, da diese über einen Touch-Screen und einen Neigungssensor verfügen. Der Roboter könnte dadurch gesteuert werden. Ziel der Bachelorarbeit soll es daher sein, den Roboter mittels Tablet-PC zu bedienen.

Abstract:

In the current industrial development, new operating concepts for robotics are becoming increasingly important. Tablet PC's offer suitable conditions for this, since they have a touch screen and a tilt sensor. The robot could be controlled by it. The aim of the bachelor thesis is therefore to operate the robot using a tablet PC.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABB	Asea Brown Boveri (Roboterhersteller)
C/S	Client/Server
IOS	Iphone Operating System
MATLAB	Matrix Laboratory
OO	Object Oriented
RAD	Rapid Application Development
SDK	Software Development Kit
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface
VCL	Visual Component Library
VFI	Visuelle Formvererbung Interface
WLAN	Wireless Local Area Networks

1 Einleitung

Mit der Entwicklung von Robotersteuerungstechnologie und Internettechnologie ist die Fernsteuerung von Robotern, die das Netzwerk verwenden, zu einer neuen Art der Fernsteuerung von Robotern geworden. Obwohl die Fernbedienung des Roboters realisiert ist, ist die Flexibilität des Steuerterminals nicht ausreichend.

Daher beginnt diese Arbeit mit der Verbesserung der Flexibilität des Steuerungsendes und kombiniert die schnell entwickelnde mobile intelligente Terminal-System-Android-Plattform mit den Robotern, entwickelt eine mobile Roboterfernsteuerungslösung über ein mobiles intelligentes Terminal, nämlich das mobile Roboterfernsteuerungssystem auf der Basis der Android-Plattform.

Diese Arbeit stellt das Designkonzept eines mobilen Roboter-Fernsteuerungssystems vor, das auf der Android-Plattform basiert, d.h. verwendet ein Mobiltelefon oder einen Tablet-Computer mit einer Android-Plattform über ein drahtloses Netzwerk, um die Fernbedienung von den Robotern zu erreichen. Dann analysiert diese Arbeit die Struktur und das Steuerungsprinzip des Fernsteuerungssystems. Das System umfasst roboterbasierter Server und Android-basierte Client. Um die Machbarkeit des oben genannten Android-basierten mobilen Roboter-Fernsteuerungssystems zu überprüfen, wurde das System noch getestet.

2 Motivation und Aufgabenstellung

2.1 Motivation

Gegenwärtig sind die meisten Steuerterminals vieler auf dem Internet basierenden mobilen Roboterfernsteuersysteme Computer. Durch die Verwendung der leistungsstarken Computerleistung wird die Fernsteuerung des Roboters gut erreicht, aber die Flexibilität der Steuerung ist immer noch unzureichend. Mit der zunehmenden Beliebtheit von mobilen Smart-Geräten (Smartphones, Tablet-Computer, usw.) wurden neue Ideen zur Verfügung gestellt, um die Flexibilität der Roboter-Fernsteuerung weiter zu verbessern, bzw. Mobile intelligente Terminals zur Fernbedienung von den Robotern.

In der aktuellen industriellen Entwicklung werden neue Bedienkonzepte für die Robotik immer wichtiger. Tablet-PC's bieten hierfür geeignete Voraussetzungen, da diese über einen Touch-Screen und einen Neigungssensor verfügen. Der Roboter könnte dadurch gesteuert werden. Ziel der Bachelorarbeit soll es daher sein, den Roboter mittels Tablet-PC zu bedienen.

In diesem Thema wird ein auf der Android-Plattform basierendes Roboterfernsteuerungssystem untersucht und entwickelt, um dessen Leistungsfähigkeit zu testen. Am Ende kann der Roboter über ein Android-Tablet über WLAN ferngesteuert werden.

2.2 konkrete Aufgaben

Bereitgestellte Hardware:

- 6-Achs-Roboter ABB IRB 120
- Tablet-PC

Die Programmierung des Tablet-PC's sollte in der Programmiersprache Delphi erfolgen.

Meilensteine:

- Einarbeitung TCP/IP-Kommunikation und WLAN
- Einarbeitung in die Programmierung am Tablet-PC sowie Erstellung von Bedienschnittstellen am Tablet-PC
- Einarbeitung in die vorhandene Sensorik im Tablet-PC
- Kommunikation zwischen Tablet-PC und PC sowie Robotersteuerung
    - Die Kommunikation sollte mittels TCP/IP (über WLAN) realisiert werden

- Achsbewegung des Roboters mittel Touch-Screen
    - Der Tablet-PC soll eine Applikation zeigen, über welche alle Achsen des Roboters bewegt werden können
- Kartesische Bewegung des Roboters mittels Touch-Screen
    - Der Tablet-PC soll eine Applikation zeigen, welche es ermöglicht, den Roboter in einem Koordinatensystem zu bewegen
- Kartesische Bewegung des Roboters mittels Neigungssensor
    - Das Neigen des Tablets bewirkt eine lineare Bewegung des Roboters

3 Technikstand und Technische Voraussetzungen

3.1 Android-Plattform

Android ist ein Betriebssystem und gleichzeitig auch eine Software-Plattform für mobile Geräte wie Smartphones, Notebooks und Tablet-PCs. Android gilt derzeit als größter Konkurrent der Apple-Plattform IOS. Das quelloffene Betriebssystem bietet verschiedene Vorteile gegenüber IOS und wird mittlerweile bei unzähligen Smartphones und Tablet-PCs eingesetzt.

Das Android-System ist ein Open-Source-Betriebssystem auf Basis des Linux-Betriebssystems des IT-Giganten Google, dessen Systemarchitektur von Grund auf in vier Ebenen unterteilt werden kann: Linux-Kernel-Layer, Bibliothek und Runtime-Layer sowie das Anwendungsprogramm.

Android Betriebssystem Vorteile[8]:

(1) Das quelloffene Betriebssystem

Android ist eine freie Software und quelloffen. Das bedeutet, dass der Quelltext der Software öffentlich zugänglich ist und von jedem Programmierer eingesehen werden kann. Die Weiterentwicklung steht jedem offen.

(2) Integrierte Google-Engine

Googles leistungsstarke Engine-Technologie wurde im Web lange Zeit angezeigt. Daher integriert das Android-Betriebssystem die Google-Engine-Technologie, um es attraktiver und wettbewerbsfähiger zu machen. Wie Google Maps, E-Mails, Suche usw. Das Android-System hat einen einzigartigen Vorteil im mobilen Smart-Terminal-Betriebssystem.

(3) Schöne UI-Schnittstellen

Android-Anwendungen werden alle mit der Programmiersprache Java entwickelt. Die Sprache Java ist führend in der Entwicklung von UI-Sprachen. Daher verwendet das Android-System Java als eine Anwendungsentwicklungssprache, die die UI-Schnittstellen der entwickelten Anwendung sehr schön macht.

(4) Kenie Entwicklungsrestriktion

Ein weiterer wesentlicher Vorteil der Android-Plattform ist die Tatsache, dass der Urheber Google nicht sehr restriktiv damit umgeht. Jeder Entwickler kann Android SDK und zugehörige Entwicklungstools herunterladen, um Android-Anwendungen zu entwickeln.

3.2 Fernsteuerungssystemimplementierung

Netzwerktechnik und Kommunikationstechnik stellen die Grundlage für die Fernsteuerung dar. Die schnelle Entwicklung der Netzwerktechnologie und Kommunikationstechnik ermöglicht eine schnelle Entwicklung der Fernsteuerungstechnologie.

3.2.1 Systemdesignideen

Die Entwurfsidee des Systems besteht darin, den Roboter über ein drahtloses lokales Netzwerk über ein mobiles Android-Tablet-PC fernzusteuern.

Das Modelldiagramm ist in der Abbildung dargestellt:

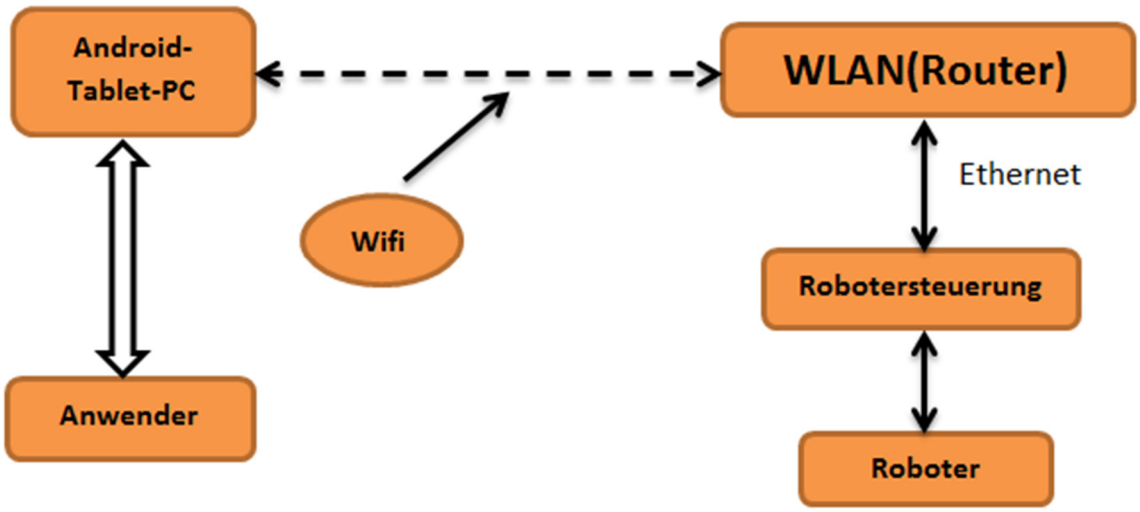

Abbildung 3.2-1: Modelldiagramm von der Entwurfsidee des Systems

3.2.2 Systemstruktur

Das Fernsteuerungssystem verwendet den C / S-Modus, d.h. den Client / Server-Modus. Das System besteht aus einem Mobile-Smart-Terminal-basierten Client und einem Roboter-basierten Server. Der Client verbindet sich über ein drahtloses lokales Netzwerk (WLAN) mit dem Server.

(1) Server

Der Server befindet sich in der Robotersteuerung. Es ist hauptsächlich dafür verantwortlich, die vom Client gesendeten Steueranweisungen zu empfangen, den Roboter zu steu-

ern, entsprechende Bewegungen auszuführen und die Roboterantwort zum Client zu senden.

(2) Client

Der Client befindet sich auf dem Android Mobile Smart-Terminal. Es ist hauptsächlich verantwortlich für das Senden von Steueranweisungen an den Server und das Empfangen von der Roboterantwort, die vom Server gesendet werden.

(3) WLAN[9]

Wireless-LAN ist die Brücke zwischen dem Client und dem Server in diesem System. Es kompensiert die Unzulänglichkeit von verkabelten Netzwerken, wodurch die Geräte, die auf das Netzwerk angewiesen sind, lokal mobil sind. Das Netzwerkverbindungsproblem wurde gelöst, das mit der verkabelten Methode nur schwer zu realisieren ist. Durch den Einsatz von WLAN können die Flexibilitätsvorteile von mobilen Smart Terminals voll genutzt werden. Der Client und der Server interagieren mit Befehlen über ein WLAN.

3.2.3 Kommunikationsmechanismus

Um die Interaktion zwischen dem Client und dem Server zu realisieren, wird in dieser Arbeit die Socket-Technologie für die Systemkommunikation verwendet. Der Client baut eine Socket-Verbindung mit dem Server über die richtige IP-Adresse auf und führt den Informationsaustausch aus. Um die Kommunikation zwischen der Android-Plattform und dem Roboter zu gewährleisten, müssen gleichzeitig die Netzwerkkommunikationsprotokolle unterstützt werden.

Das TCP/IP-Protokoll stellt derzeit ein weit verbreitetes Internetprotokoll dar. Dank der Benutzerfreundlichkeit, der Portabilität und Entwicklungsperspektiven ist die Entwicklung von Kommunikationssoftware unter Verwendung des TCP/IP-Protokolls eine optimale Wahl. Die leistungsfähige Funktion kann die Verbindung und die Kommunikation des Netzes verwirklichen. Die wichtigsten Teile des TCP/IP-Protokolls sind TCP, IP, UDP und Physical Interface. In einem Netzwerkbetriebssystem, das das TCP/IP-Protokoll verwendet, können verschiedene Anwendungen mit den zugrunde Kernprotokollen über denselben Netzwerkprogrammierschnittstellen-Socket kommunizieren. Es ermöglicht eine Anwendung, eine Verbindung zu einer andere Anwendung auf einer anderen Maschine herzustellen, ohne die Details des zugrunde Protokolls berücksichtigen zu müssen. Basierend auf der obigen Analyse, kombiniert mit den Entwicklungsbedingungen des Fernsteuerungssystems und der Skalierbarkeit des Systems, wurde das TCP/IP-Protokoll als

Kommunikationsprotokoll für das entwickelte Roboter-Fernsteuerungssystem ausgewählt. [11], [15]

Die IP-Adressen der Geräte sind in der Tabelle dargestellt:

Gerät	IP-Adresse	Subnetzmaske
Robotersteuerung	141.55.217.212	255.255.248.0
Android-Tablet-PC	141.55.217.213	255.255.248.0
Router	141.55.217.214	255.255.248.0

Tabelle 3.2-1: IP-Adresse und Subnetzmaske der Geräte

3.2.4 Zugrunde liegende Interaktion

Der Server des Systems ist in ABBScript-Sprache geschrieben, der Client wird durch Delphi verwirklicht und die Robotersteuersimulation wird in Matlab realisiert .

Die Systemsoftware-Entwicklungsumgebung und Kerntechnologien sind wie folgt:

1) Betriebssystem: Windows 7,
2) Kerntechnologie: Delphi, ABBScript, MatlabScript
3) Entwicklungstools: Matlab R2015a, RAD Studio 10.2
4) Entwicklungstoolspackage: Android SDK 24.3.3

Die Ferninteraktion zwischen dem Client und dem Server erfordert das Übergeben von Anweisungen untereinander. Daher ist es vor der Entwicklung des Systems notwendig, das Steuerprotokoll zu bestimmen, d.h. die Befehle, die der Server und der Client miteinander kommunizieren und erkennen können. Um die Entwicklung zu vereinfachen, wurde die Programmierer zunächst die Steuerbefehle, die die Robotersteuerung interpretieren kann, untersucht.

3.3 Systemserver-Entwicklung mit ABBScript

Die Rolle des Servers besteht darin, auf die Verbindungsanforderung des Clients zu antworten und eine Verbindung herzustellen. Der Server akzeptiert die Steueranweisungen des Clients, interagiert mit dem Robotersteuerungsprogramm, um die Bewegung des Roboters zu steuern und sendet die Roboterantwort zum Client.

Der Server des Systems ist in ABBScript-Sprache geschrieben.

Einige Befehle in ABBScript-Sprache sind in der Tabelle 3.2-2 und Anlage-ABBScript-Sprache dargestellt:

set_digital_output	setzt ein neue Ausgang	
movej	Achsbewegung	
movel	Linearbewegung	zu einem festen Punkt
		zu einem Offset eines festen Punktes
		zu einem Offset der aktuellen Position
exit	beendet den Verlauf	

Tabelle 3.2-2: Einige nutzbare Befehle in ABBScript-Sprache

3.4 Systemclient-Entwicklung mit Delphi

Der Client wird unter der Android-Plattform entwickelt, verwendet Socket, um eine Verbindung zum Server herzustellen. Wenn der Client und der Server eine Verbindung herstellen, kann der Anwender über den Client die Steueranweisungen an den Server senden und die Roboterantwort empfangen, die vom Server gesendet werden.

Delphi, eingeführt von Borland, ist eine neue visuelle Programmierumgebung, die ein praktisches und schnelles Windows-Anwendungsentwicklungstool bietet. Es verwendet viele der erweiterten Funktionen und Designkonzepte der grafischen Benutzeroberfläche von Microsoft Windows und verwendet eine flexible, wiederverwendbare objektorientierte Programmiersprache, den weltweit schnellsten Compiler und führende Datenbanktechnologie. Für die meisten Programmentwickler wird der Einsatz von Delphi zur Entwicklung von Anwendungssoftware die Programmierungseffizienz zweifellos erheblich verbessern.

Die Eigenschaft von Delphi:

(1) Visuelle Entwicklungsumgebung

Die visuelle Entwicklungsumgebung ist normalerweise in drei Komponenten unterteilt: einen Editor, einen Debugger und einen Formulardesigner. Wie die meisten modernen RAD-Werkzeuge (Rapid Application Development) arbeiten diese drei Teile zusammen. Wenn die Anwender im Formular-Designer arbeiten, generiert Delphi automatisch einen Code für die Steuerelemente, die die Anwender im Formular bearbeiten. Die Anwender können dem Editor auch Code hinzufügen, um das Verhalten der Anwendung zu definieren. Die Anwender können das Programm auch debuggen, indem die Anwender Haltepunkte, Überwachungspunkte usw. im selben Editor festlegen.

Der Delphi-Formular-Designer unterscheidet sich dadurch, dass Delphi auf einem wirklich objektorientierten Framework basiert. Auf diese Weise werden alle Änderungen, die die Anwender an der Basisklasse vornehmen, an alle abgeleiteten Klassen übergeben. Eine der Schlüsseltechnologien, die hier involviert sind, ist VFI (visuelle Formvererbung). Mit der VFI-Technologie können die Anwender jedes andere Formular in der aktuellen Pro-

jekt- oder Objektbibliothek dynamisch erben. Sobald sich das Basisformular ändert, wird das abgeleitete Formular sofort aktualisiert.

(2) Compiler-Geschwindigkeit und Effizienz des kompilierten Codes

Der schnelle Compiler ermöglicht es, nach und nach Software zu entwickeln, den Quellcode zu modifizieren, zu kompilieren, zu testen, zu modifizieren, neu zu kompilieren und erneut zu testen. Wenn die Kompilierungsgeschwindigkeit langsam ist, müssen Entwickler den Code in Stapeln ändern und vor jeder Kompilierung mehrere Änderungen vornehmen, um einen effizienten Zyklus zu ermöglichen. Es ist selbstverständlich, dass die Vorteile der Verbesserung der Betriebseffizienz, der Einsparung von Betriebszeit und der Erzeugung von Binärcode noch kürzer sind.

(3) Die Funktion und Komplexität von Programmiersprachen

Object Pascal hat ein gutes Verständnis für die Balance von Komplexität und Funktionalität und begrenzt die verfügbaren Funktionen, um das logische Design des Entwicklers zu verbessern. Delphi vermeidet die Vorstellung von Mehrfachvererbung, die vollständig objektorientiert ist, aber leicht missbraucht werden kann, und implementiert stattdessen eine Klasse, die mehrere Schnittstellenfunktionen implementiert. Deshalb verfügt Delphi über einige leistungsstärke Funktionen wie Ausnahmebehandlung.

(4) Datenbankstruktur Flexibilität und Skalierbarkeit

Da Borland keinen Datenbankplan hat, behält Delphi die unter allen Tools als die flexibelste Datenbankstruktur bei. Für die meisten Anwendungen, die auf lokalen, Client / Server- und ODBC-Datenbankplattformen (Open Database Connectivity) basieren, sind BDEs (Borland Database Engine) sehr leistungsfähig.

(5) Framework für die Erweiterung von Design- und Nutzungsmustern

Dies ist ein wichtiges Merkmal, das von anderen Softwaredesigntools häufig übersehen wird. VCL (Visual Component Library) ist der wichtigste Teil von Delphi. Die Fähigkeit, Komponenten zur Entwurfszeit zu manipulieren, Komponenten zu erstellen und die OO-Technologie (Object Oriented) zu verwenden, um das Verhalten anderer Komponenten zu übernehmen, ist ein Schlüsselfaktor für die Effizienz von Delphi. In vielen Fällen verwendet das Schreiben von VCL-Komponenten einen festen OO-Entwurfsansatz. Im Gegensatz dazu sind andere komponentenbasierte Frameworks oft zu starr oder zu komplex. [13]

4 Bedienschnittstelle und Steueralgorithmus am Tablet-PC

Die verschiedenen Bewegungen des Roboters werden von den verschiedenen Gelenkachsen durchgeführt, die Robotersteuerung steuert im Wesentlichen die einzelne Gelenkachse. Jede Achse hat einen Freiheitsgrad, der die Rotationsaufgabe in eine bestimmte Richtung vollenden kann, wobei alle Achsen mit koordinierter Bewegung relativ komplexe Bewegungen gleichzeitig ausführen können. Die Aufgabe der Motion-Control-System-Software besteht in der Steuerung dieser Gelenkachsen, die durch den zugrunde liegenden Motion Controller realisiert werden.

Die mit Delphi erstellte Anwendungssoftware des Robotersteuerungssystems umfasst hauptsächlich die Bedienschnittstelle, die Parametereinstellung, die Multifunktionsrealisierung und die Aufgabenausführung.

4.1 Matlabschnittstelle

Bevor jedes Programm getestet in den Roboter wird, kann die Simulation in Matlab durchgeführt werden. Die Bedienschnittstelle der Simulation in Matlab ist in der Abbildung 4.1-1 dargestellt:

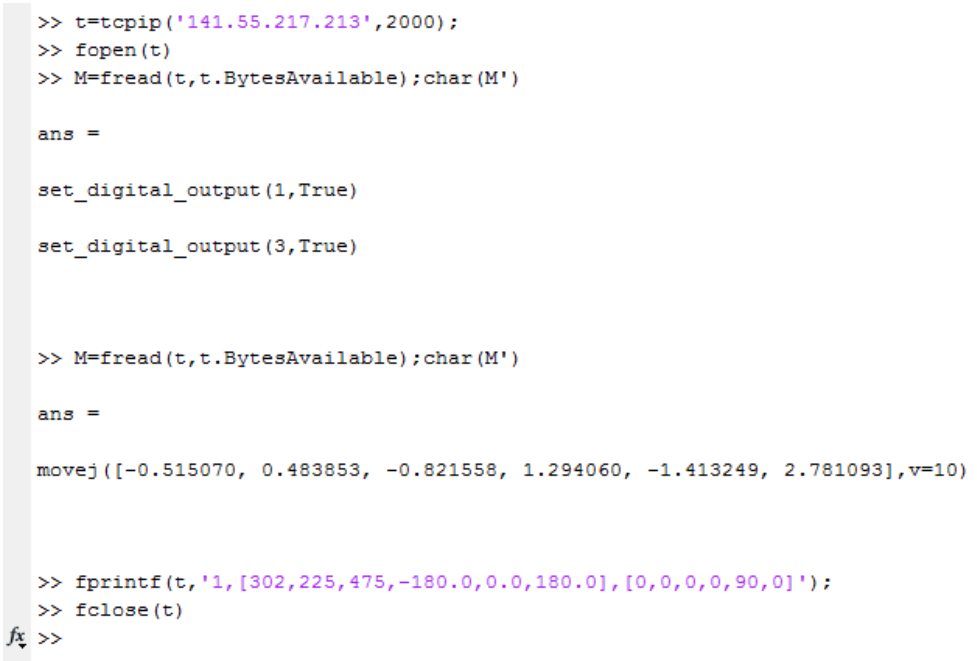

Abbildung 4.1-1: Die Bedienschnittstelle der Simulation in Matlab

Die Schritt-für-Schritt Analyse ist wie folgt:

$t = tcpip('141.55.217.213', 2000);$
// Anschluss Aufbau :IP-Adresse vom Android-Tablet-PC und Port_2000 vom Roboter
$fopen(t)$
// Port_2000 anschalten
$M = fread(t, t.BytesAvailable); char(M')$
// die Daten nach Bytes aus der Binärdatei ablesen und in String-Array konvertieren
$fprintf(t, '1, [302,225,475, -180.0,0.0,180.0], [0,0,0,0,90,0]');$
// die Daten über die Roboterantwort in eine Textdatei einschreiben
(ein Beispiel: Position_XYZ + Eulersche Winkel + 6 Achswinkel)
Es ist die Simulation des Empfangs von der Roboterantwort
$fclose(t)$
// Port_2000 schließen

4.2 Delphi-Programmiersprache

4.2.1 Einige verwendeten Delphi-Datentyp-Konvertierungsfunktionen

Einige Delphi-Datentyp-Konvertierungsfunktionen sind in der Tabelle dargestellt:

Datentyp-Konvertierungsfunktionen	Beschreibung
IntToStr	konvertiert einen Integerwert in einen String
StrToInt	konvertiert einen String, der einen Integerwert repräsentiert (in dezimaler oder hexadezimaler Notation), in eine Zahl
StrToFloat	konvertiert einen angegebenen String in einen Gleitkommawert
FloatToStr	konvertiert eine Gleitkommazahl in einen String
FloatToStrF	konvertiert die in Value angegebene Gleitkommazahl in die entsprechende String-Darstellung

Tabelle 4.2-1: Einige verwendeten Delphi-Datentyp-Konvertierungsfunktionen

4.2.2 Einige verwendeten Delphi-Datenverarbeitungsfunktionen

Einige verwendete Delphi-Datenverarbeitungsfunktionen sind in der Tabelle dargestellt:

Datenverarbeitungsfunktionen	Beschreibung
Pos-Funktion	sucht einen Teilstring in einem angegebenen String Das Rückgabeergebnis von $pos\ ('b', abcd\ )$ ist 2
Copy-Funktion	gibt einen Teilstring eines Strings oder ein Segment eines

	dynamischen Arrays zurück Das Rückgabeergebnis von $S:=copy\ ('delphi', 4, 2)$ ist $'ph'$
StringReplace-Funktion	ersetzt Vorkommen eines Teilstrings in einem String Das Rückgabeergebnis von StringReplace $('not\ a\ pen', 'a', 'two', [])$ ist $'not\ two\ pen'$

Tabelle 4.2-2: Einige verwendeten Delphi-Datenverarbeitungsfunktionen

4.2.3 TIdTCPServer-Modul-Anwendung

IdTCPServer ist eine Gruppe von Open-Source-Internet-Steuerung, die die meisten gängigen Internet-Protokolle unterstützt.

IdTCPServer enthält einen vollständigen TCP-Server mit mehreren Threads. Das Steuerelement verwendet einen oder mehrere Threads zum Abhören von Clientverbindungen.In Verbindung mit *TIdThreadMgr* wird jeder Thread einer Verbindung mit dem Client zugewiesen. Nach dem Start von *IdTCPServer* wird automatisch ein Überwachungsthread *TIdListenerThread* eingerichtet, der für die Überwachung von Clientverbindungsanforderungen zuständig ist und einen *TIdPeerThread-Thread* für jede vom Server akzeptierte Verbindung erstellt. Jede Verbindung implementiert eine Dateninteraktion mit dem Server, indem sie ihr eigenes *TIdPeerThread* ausführt.

IdTCPServer ist Blocking, d.h. wenn im Blockierungsmodus keine Verbindungsanforderung eingeht, wird das Warten auf den Verbindungsaufruf blockiert, bis eine Verbindungsanforderung eintrifft. Winsock oder Windows Sockets ist eine Windows-Programmierschnittstelle (API) zum Zugriff auf Rechnernetze über Sockets. Wenn Winsock verwendet wird, um eine Webanwendung zu entwickeln, wird das Lesen von Daten aus einem Socket oder das Schreiben von Daten in einen Socket asynchron ausgeführt, sodass die Ausführung von anderem Code im Programm nicht blockiert wird. Beim Empfang von Daten sendet Winsock die entsprechende Nachricht an die Anwendung. Diese Art von Zugriff wird als nicht blockierende Verbindung bezeichnet, bei der Sie auf Ereignisse reagieren müssen, die Statusmaschine einrichten müssen und normalerweise auch eine Warteschleife benötigen.

Das Diagramm_IdTCPServer ist wie folgt:

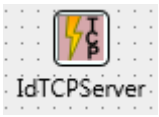

Abbildung 4.2-1: Diagramm des TIdTCPServer-Moduls

Relevante Eigenschaften:

1. Active

Wenn dieser Eigenschaftswert *True* ist, wird der Server aktiviert und wartet darauf, dass der Client eine Verbindung mit ihm herstellt. Wenn der Wert dieser Eigenschaft *False* ist, wird der Server heruntergefahren und alle Verbindungen werden freigegeben.

2. Bindings

Dieses Attribut ist der Container, der das vom Server zugewiesene Socket-Handle enthält. Wenn der Client eine Verbindung mit dem Server herstellt, wird die IP-Adresse des Clients auf dem Server angezeigt.

3. DefaultPort

Dieses Attribut wird verwendet, um die Portnummer anzugeben, die vom Server für die Verbindung mit dem Client verwendet wird.

Relevante Ereignisse:

Das Modul wird verwendet, um die Verbindung zwischen den zwei Maschinen zu erreichen, und unterstützt die folgenden Ereignisse:

1. OnConnect

Dieses Ereignis wird ausgelöst, wenn ein Client eine Verbindung zum Server herstellt.

2. OnDisconnect

Dieses Ereignis wird ausgelöst, wenn ein Client vom Server getrennt wurde.

3. OnExecute

Wenn der Server eine neue Verbindung mit dem Client herstellt, richtet der Server einen Thread für diese Verbindung ein. Dieses Ereignis wird ausgelöst, wenn der Thread ausgeführt wird.

4. OnException

Dieses Ereignis wird ausgelöst, wenn die daten nicht korrekt gelesen und geschrieben werden.

4.2.4 TMemo-Modul-Anwendung

TMemo ist eine mehrzeilige Texteingabe-Komponente zum Platzieren eines mehrzeiligen Texteingabebereichs in einem Fenster. Durch die Verwendung der *TMemo*-Komponente kann dem Benutzer ein Eingabebereich für eine große Menge an Informationen zur Verfügung gestellt werden, der mehrere Megabyte an Daten enthalten kann.

Das Diagramm_Memo ist wie folgt:

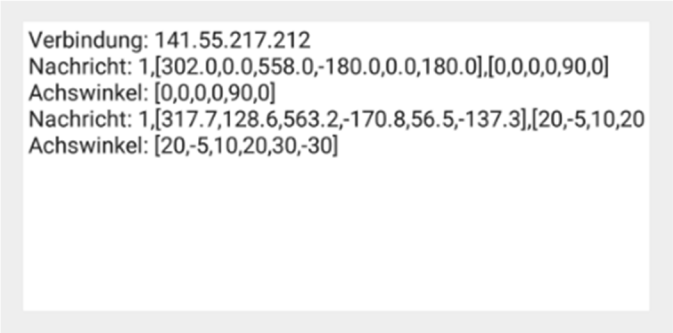

Abbildung 4.2-2: Diagramm des TMemo-Moduls

Relevante Eigenschaften:

1. Lines

Die Benutzer verwenden diese Eigenschaft, um den Text des Texteditors in Zeileneinheiten zu bearbeiten. Diese Eigenschaft ist ein *TStrings*-Objekt, sodass die *TStrings*-Methode zum Ausführen verschiedener Operationen verwendet werden kann, so wie die Berechnung die Anzahl der Textzeilen, das Hinzufügen einer Zeile, das Löschen einer Zeile und das Ersetzen einer Zeile durch neuen Text.

Anwendung:

Wenn dem Memo-Feld im Programm Daten hinzugefügt werden müssen, können die entsprechenden Methoden der Eigenschaft Lines verwendet werden.

Der folgende Code implementiert die Verwendung der Eigenschaft Lines, um Daten zu einer Memo-Box hinzuzufügen.

```
procedure TForml.ButtonlClick(Sender: TObjcct);
begin
Memol Lines.Add( ' Hallo ' );
end;
```

2. EnabledScroll

Mit dieser Eigenschaft wird festgelegt, ob einem mehrzeiligen Texteditor eine horizontale oder vertikale Bildlaufleiste hinzugefügt werden soll. Wenn dieser Eigenschaftswert *True* ist, gibt es beide Bildlaufleisten horizontal und vertikal.

Relevante Methoden:

1. Clear

Diese Methode wird verwendet, um den gesamten Text in der Bearbeitungskomponente zu löschen, dabei wird der Text durch ein Nullzeichen ersetzt.

Anwendung:

```
Memo1.Clear;
```

4.2.5 TButton-Modul-Anwendung

TButton bietet Delphi-Programomierern die Funktionalität von Befehlsschaltflächen. Befehlsschaltflächen sind in Fenstern und Dialogfenstern von Windows üblich und bestehen aus Rechtecken, die eine Textbeschriftung enthalten, um bestimmte Arten von Operationen zu aktivieren. Das häufigste Ereignis für die Befehlsschaltfläche ist das *OnClick*-Ereignis, das ausgelöst wird, wenn der Benutzer auf die Schaltfläche klickt.

Das Diagramm_Button ist wie folgt:

Abbildung 4.2-3: Diagramm des TButton-Moduls

Relevante Eigenschaften:

1. Text

Mit dieser Eigenschaft wird der Name des Buttons festgelegt.

2. Visible

Mit dieser Eigenschaft steuert die Sichtbarkeit.

Relevante Methoden:

1. Click

Die *Click*-Methode wird verwendet, um einen Mausklick zu simulieren.

Relevante Ereignisse:

1. OnClick

Das *OnClick*-Ereignis ausgelösen, wenn Benutzer den Button klickt.

Beim Entwerfen eines Programms ist das *OnClick*-Ereignis fast immer erforderlich, wenn der Benutzer eine Schaltflächenkomponente verwendet. Wenn der Benutzer auf die Schaltfläche klickt, ruft das Programm den ‚Ereignishandler der Schaltfläche auf.

Der Code ist wie folgt:

```
procedure TForml.ButtonlClick(Sender: TObjcct);
begin
Memo1.Lines.Clear;
end;
```

4.2.6 TImage-Modul-Anwendung

Mit der *TImage*-Komponente wird ein Bild in einem Formular angezeigt.

Das Diagramm_Image ist wie folgt:

Abbildung 4.2-4: Diagramm des TImage-Moduls

Relevante Eigenschaften:

1. MultiResBitmap

Dieses Eigenschaft gibt das in der Imagekomponente angezeigte Bild an, enthält ein Symbol, eine Metadatei, ein Bitmap-Bild oder ein benutzerdefiniertes Bild.

4.2.7 TScrollbar-Modul-Anwendung

TScrollbar-Komponente werden verwendet, um Fenster oder Komponenteninhalte zu scrollen. Bei der Verwendung kann bestimmte Steuerelemente einfach manipuliert werden.

Das Diagramm_Scrollbar ist wie folgt:

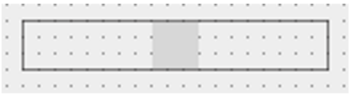

Abbildung 4.2-5: Diagramm des TScrollbar-Moduls

Relevante Eigenschaften:

1. Max

Diese Eigenschaft wird verwendet, um den maximalen Wert der Bildlaufleiste festzulegen.

2. Min

Diese Eigenschaft wird verwendet, um den Mindestwert der Bildlaufleiste festzulegen.

3. SmallChange

Die Bildlaufleiste scrollt in einer bestimmten Entfernung, wenn der Benutzer die Pfeile an den Enden der Bildlaufleiste drückt.

4. Value

Diese Eigenschaft wird verwendet, um den Anfangswert der Bildlaufleiste festzulegen.

Relevante Ereignisse:

1. Onchange

Dieses Ereignis wird ausgelöst, wenn der Benutzer die Bildlaufleiste bedient oder den Wert der Eigenschaft Position direkt ändert.

Der Code ist wie folgt:

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
Label1.Text := FloatToStrF(scrollbar1.Value, ffFixed, 8, 6);
end;
```

4.2.8 TEdit-Modul-Anwendung

TEdit kapselt die Funktionalität der meisten Standard-Win32-Bearbeitungskomponenten, die manchmal als Textfelder bezeichnet werden. Das Bearbeitungsfeld enthält eine einzelne Textzeile, die der Benutzer bearbeiten oder ändern kann. Der Text im Bearbeitungsfeld kann nach Bedarf als schreibgeschützt festgelegt werden, der von der Benutzer nicht verändert werden kann.

Das Diagramm_Edit ist wie folgt:

Abbildung 4.2-6: Diagramm des TEdit-Moduls

Relevante Eigenschaften:

1. MaxLength

Der Programmier verwenden diese Eigenschaft, um die Anzahl der vom Benutzer eingegebenen Zeichen einzuschränken, und wenn der Wert 0 ist, ist für die Länge des Zeichens keine benutzerdefinierte Begrenzung definiert. Die Benutzer verwenden diese Eigenschaft, um die Textlänge in der Bearbeitungskomponente zu begrenzen, wenn sie den Text in der Bearbeitungskomponente in einen Puffer fester Länge kopieren möchten. Die Eigenschaft wird verwendet, um die maximale Anzahl der zulässigen Zeichen im Textfeld festzulegen.

2. Text

Die *Text*-Eigenschaft wird verwendet, um den Text in dem Bearbeitungsfeld zu lesen oder festzulegen.

Relevante Ereignisse:

1. OnChange

Dieses Ereignis wird ausgelöst, wenn sich der Text in der Bearbeitungskomponente ändert, und sein Ereignishandler ist so programmiert, dass er eine bestimmte Funktion ausführt. Das *OnChange*-Ereignis stellt die erste Antwort für den Benutzer dar, der eine Benachrichtigung über die Bearbeitungskomponente eingibt.

Anwendung:

Das Ereignis wird ausgelöst, wenn sich der Inhalt im Textfeld ändert. Die Anwender verwenden dieses Ereignis, um den Benutzer aufzufordern, legitime Daten einzugeben.

Der Code ist wie folgt:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
Scrollbar1.Value :=StrToFloat(Edit1.Text);
end;
```

4.2.9 TCheckBox-Modul-Anwendung

Die CheckBox-Komponente stellt dem Benutzer ein Kontrollkästchen zur Verfügung. Es kann ausgewählt oder nicht ausgewählt werden. Die Benutzer können klicken, um auszuwählen und nochmal klicken, um die Auswahl aufzuheben.

Das Diagramm_CheckBox ist wie folgt:

Abbildung 4.2-7: Diagramm des TCheckBox-Moduls

Relevante Eigenschaften:

1. Text

Mit dieser Eigenschaft wird der Name der CheckBox festgelegt.

2. Check

Diese Eigenschaft ist *True*, wenn das Kontrollkästchen aktiviert ist. Diese Eigenschaft ist *False*, wenn das Kontrollkästchen deaktiviert oder grau ist.

Relevante Ereignisse:

1. OnChange

Das Ereignis wird ausgelöst, wenn sich der Zustand des Kontrollkästchens ändert.

Anwendung:

Der Code ist wie folgt:

```
procedure TForm1.CheckBox3Change(Sender: TObject);
begin
if CheckBox3.IsChecked=True then hContext.Connection.IOHandler.WriteLn( '
set_digital_output(3,True) ' ) ;
end;
```

4.2.10 TRadioButton-Modul-Anwendung

Die *TRadioButton*-Komponente kapselt die Windows-Optionsschaltfläche, die manchmal als Optionsfeld bezeichnet wird. Optionsfelder sind in Windows üblich und erscheinen als kreisförmiger, externer Block mit Textbeschriftungen. Die Optionen in einer Gruppe von Optionsfeldern schließen sich gegenseitig aus, wobei immer nur ein Optionsfeld ausgewählt werden kann.

Das Diagramm_RadioButton ist wie folgt:

Abbildung 4.2-8: Diagramm des TRadioButton-Moduls

Relevante Eigenschaften:

1. Text

Mit dieser Eigenschaft wird der Inhalt und die Funktion des RadioButtons zeigt.

2. IsChecked

Diese Eigenschaft wird verwendet, um festzustellen, ob das Optionsfeld ausgewählt ist. Wenn der Wert auf *True* gesetzt ist, wird diese Komponente ausgewählt, und andere Komponenten in derselben Gruppe werden in den nicht ausgewählten Status zurückgesetzt. Wenn der Wert auf *False* gesetzt ist, wird diese Komponente nicht ausgewählt.

3. GroupName

Der Anwender sortiert RadionButton nach dem Verwendungszweck. Der gleiche Typ wird zusammen kombiniert.

Relevante Ereignisse:

1. OnClick

Das *OnClick*-Ereignis wird ausgelöst, wenn der Benutzer der RadioButton klickt.

Anwendung:

Wenn der Benutzer auf die Schaltfläche klickt, ruft das Programm den *OnClick*-Ereignishandler der Schaltfläche auf.

Der Code ist wie folgt:

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
gesch:= ' 10 ' ;
end;
```

4.2.11 TTimer-Modul-Anwendung

Diese Komponente kapselt die Windows-Timerfunktion ein.

Das Diagramm_Timer ist wie folgt:

Abbildung 4.2-9: Diagramm des TTimer-Moduls

Relevante Eigenschaften:

1. Enable

Mit dieser Eigenschaft wird gesteuert, ob die *TTimer*-Komponente regelmäßig *OnTimer*-Ereignisse generiert. Wenn der Benutzer die Enabled-Eigenschaft auf *True* setzt, generiert die *TTimer*-Komponente periodische *OnTimer*-Ereignisse, wenn die Eigenschaft auf *False* gesetzt ist, hört der Timer auf zu generieren.

2. Interval

Diese Eigenschaft legt die Zeitdauer in Millisekunden fest, die das *OnTimer*-Ereignis auslöst, um verschiedene Funktionen zu erreichen, wie z.B. Zählt nach Sekunden, Zählt nach Minuten usw.

Relevante Ereignisse:

1. OnTimer

Das *OnTimer*-Ereignis wird einmal innerhalb einer bestimmten Zeit ausgelöst. Dieses Zeitintervall wird durch das Interval-Attribut bestimmt.

Anwendung:

Das *OnTimer*-Ereignis stellt den wichtigsten Teil der *TTimer*-Komponente dar. Der Code in diesem Ereignis führt regelmäßig eine spezielle Funktion aus. Es ist möglich, mit dem Timer zeiten zu messen und diese später auszulesen und auszuwerten. Der Code ist wie folgt:

```
procedure TForm1.Timer1Timer(Sender TObicct);
begin
Button1Click(self);
end;
```

4.2.12 TMotionSensor-Modul-Anwendung

Das Diagramm_ MotionSensor ist wie folgt:

Abbildung 4.2-10: Diagramm des TMotionSensor-Moduls

Relevante Eigenschaften:

1. Active

Wenn dieser Eigenschaftswert *True* ist, wird der Parameter des MotionSensors wie z.B. *AccelerationX* beliebig aufgeruft. Wenn dieser Eigenschaftswert *False* ist, hat der MotionSensor darauf keinen Einfluss.

Die Anwendung des MotionSensors:

Die meisten Android-Geräte verfügen über integrierte Sensoren zur Messung von Bewegung, Richtung und verschiedenen Umgebungsbedingungen. Diese Sensoren können eine Vielzahl von hochpräzisen Rohdaten bereitstellen. Diese Daten sind sehr nützlich, wenn der Benutzer die Bewegung oder Position von 3D-Geräten überwachen oder Änderungen in der Umgebung überwachen möchte. Wenn die Leute zum Beispiel ein Spiel spielen, müssen die Designer möglicherweise die Daten von neutralen Sensoren lesen, um unsere komplexen Körperhaltungen und Bewegungen in Neigung, Vibration, Rotation oder Amplitude abzuleiten. Auf ähnliche Weise kann eine Wetteranwendung die Verwendung der Temperatur- und Feuchtigkeitssensoren des Geräts erfordern. Außerdem muss eine Reiseanwendung möglicherweise Magnetsensoren und Beschleunigungssensoren verwenden, um den Kompasszeiger zu berechnen.

Einige dieser Sensoren sind hardwarebasiert und andere sind softwarebasiert. Hardwarebasierte Sensoren sind physikalische Geräte, die in Mobiltelefone oder Tablets eingebaut sind und deren Daten direkt aus der Messung bestimmter physikalischer Eigenschaften wie Beschleunigung, Erdmagnetfeldstärke oder Winkelgeschwindigkeit abgeleitet werden. Softwarebasierte Sensoren haben keine physischen Komponenten und ahmen hardwarebasierte Sensoren nach. Softwarebasierte Sensoren erfassen Daten von einem oder mehreren hardwarebasierten Sensoren.

Alle Bewegungssensoren geben Sensordaten in mehrdimensionalen Arrays im Sensor-Ereignis zurück. Diese Daten werden in den Parametern als Float-Array zurückgegeben. Zum Beispiel gibt der Beschleunigungsmesser in einem Sensorereignis Beschleunigungsdaten auf der dreidimensionalen Achse zurück, und das Gyroskop gibt die Rotationsgeschwindigkeitsdaten auf der dreidimensionalen Achse zurück.

MotionSensor stellt Informationen über Beschleunigung, Winkel, Status und Geschwindigkeit der Gerätebewegung bereit.

$AccelerationX$, $AccelerationY$ und $AccelerationZ$ geben die Beschleunigung in cm/s^2 für die X-, Y- und Z-Achse zurück.

$AngleAccelX$, $AngleAccelY$ und $AngleAccelZ$ geben die Winkelbeschleunigung in $°/s^2$ für die X-, Y- und Z-Achse zurück.

$Speed$ ermittelt die Geschwindigkeit des Gerätes in Meter pro Sekunde (m/s).

$Motion$ ermittelt, ob das Gerät aktuell bewegt wird oder nicht.

Sensordaten können sich mit hoher Geschwindigkeit ändern, was bedeutet, dass das System häufig die Methode *onStateChanged* (SensorEreignis) aufruft. Als bewährte Methode sollten die Benutzer in der *onStateChanged* (SensorEreignis) so wenig wie möglich verwenden, um den Rückruf nicht zu blockieren. Wenn die Anwendung etwas zu tun ist, sollte der Benutzer diese Arbeit außerhalb der Methode *onStateChanged* (SensorEreignis) ausführen.

MotionSensor verwendet ein Standard-Sensorkoordinatensystem.

Abbildung 4.2-11: Das Koordinatensystem des Sensors in Tablet

Vergleich des Koordinatensystems zwischen Sensor und Roboter (Siehe 4.4)

Wenn das Gerät in der Standardausrichtung platziert wird, bezieht sich das Koordinatensystem auf den Bildschirm des Geräts. Die X-Achse ist horizontal und zeigt nach rechts,

die Y-Achse ist vertikal und zeigt nach oben, und die Z-Achse steht senkrecht auf dem Display.

Das Koordinatensystem des Sensors ändert sich nicht, wenn sich das Gerät bewegt. An diesem Punkt ist der Roboter genau so wie der Sensor.

Wenn der Bediener mit dem Tablet vor dem Roboter steht, ergibt es sich folgende Beziehung:

$$Sensor\_X+ = Roboter\_Y +$$

$$Sensor\_Y+ = Roboter\_X -$$

Der Sensor misst die auf den Sensor ausgeübte Kraft und erfasst die Beschleunigung des Geräts gemäß der festgelegten Beziehung. Die Schwerkraft wirkt sich jedoch immer auf die Genauigkeit der Messung in der Beziehung aus. Um die wahre Beschleunigung der Vorrichtung zu messen, muss daher die Schwerkraftstörung in den Beschleunigungsmesserdaten ausgeschlossen werden. Deshalb werden $AccelerationX$ und $AccelerationY$ benutzt werden.

```
Label2.Text := FloatToStrF(Motionsensor1.Sensor.AccelerationX,ffFixed, 8, 2);
Label3.Text := FloatToStrF(Motionsensor1.Sensor.AccelerationY,ffFixed, 8, 2);
```

Also gibt es die folgenden Schlussfolgerungen:

$$Senosr\_AccelerationX + \longrightarrow Roboter\_VerschiebungY +$$
$$Senosr\_AccelerationX - \longrightarrow Roboter\_VerschiebungY -$$
$$Senosr\_AccelerationY + \longrightarrow Roboter\_VerschiebungX -$$
$$Senosr\_AccelerationY - \longrightarrow Roboter\_VerschiebungX +$$

4.3 Achsbewegung des Roboters mittel Touch-Screen

Das ist eine Applikation, über welche alle Achsen des Roboters bewegt werden können.

Durch Einstellung von 6 Achswinkeln und Geschwindigkeit kann der Roboter eine beliebige Position erreichen.

Das Koordinatensystem des Roboters in Kartesische Bewegung ist in der Abbildung 4.3-1 dargestellt:

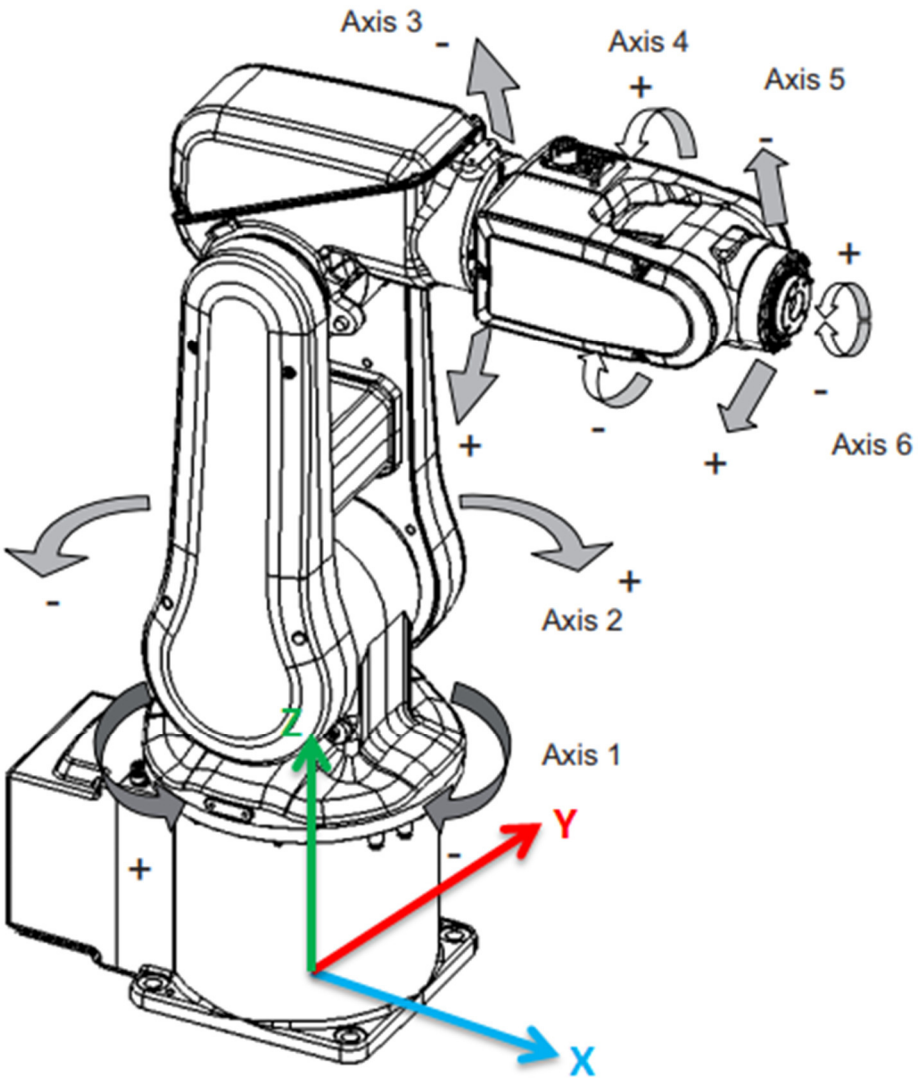

Abbildung 4.3-1: Das Koordinatensystem des Roboters (Quelle: [6, Seite11])

4.3.1 Bedienschnittstellendesign

Als eine Bedienapplikation ist nicht nur ein effizienter und einfacher Algorithmus wichtig. Eine ästhetische freundliche Bedienschnittstelle ist auch unerlässlicher Bestandteil des Designs.

Das Bedienschnittstellendesign der ersten Teilaufgabe auf der Android-Ansicht ist in der Abbildung 4.3-2 dargestellt:

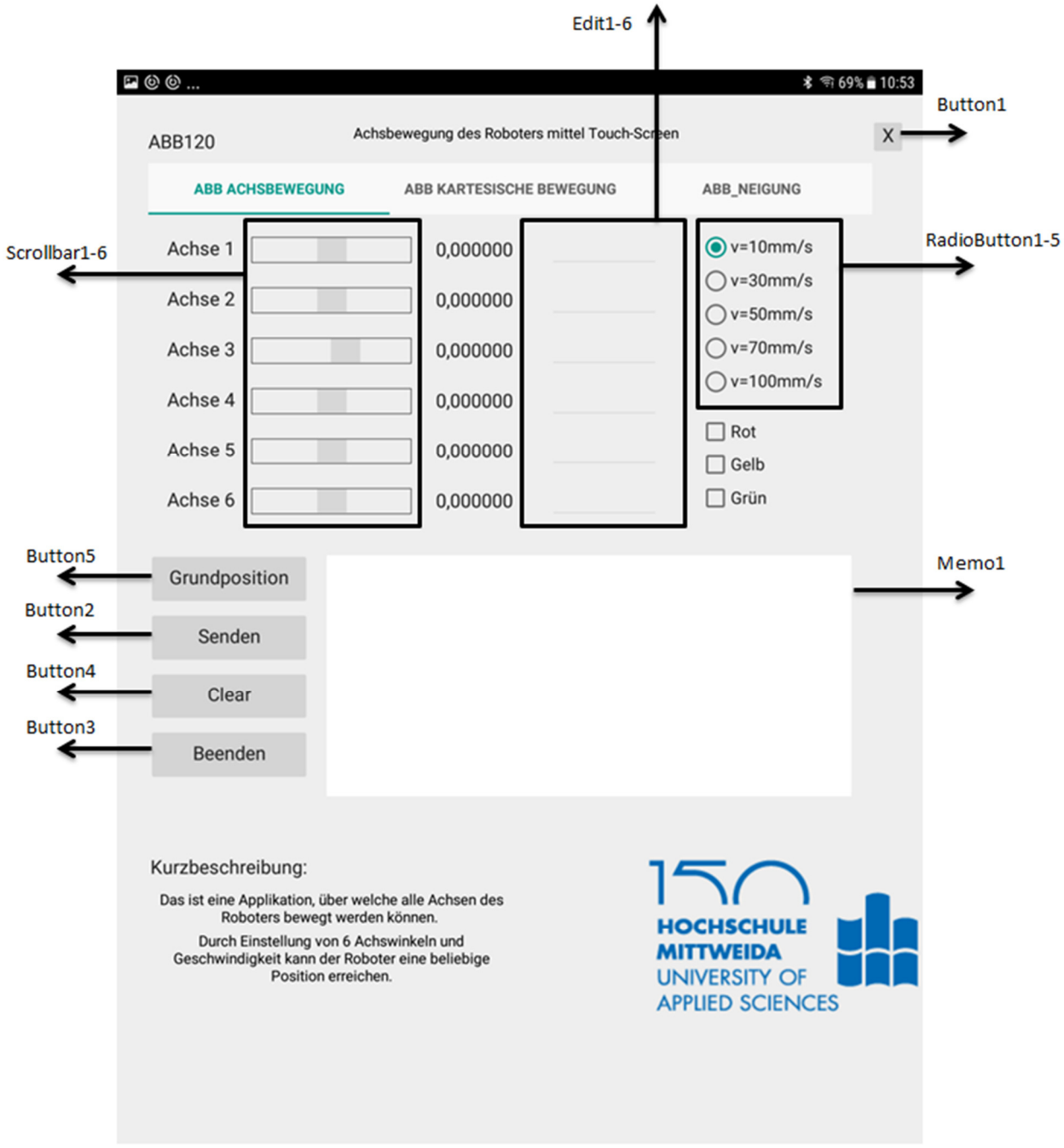

Abbildung 4.3-2: Das Bedienschnittstellendesign_1 auf der Android-Ansicht

Das Bedienschnittstellendesign der ersten Teilaufgabe auf der Delphi-Ansicht ist in der Abbildung 4.3-3 dargestellt:

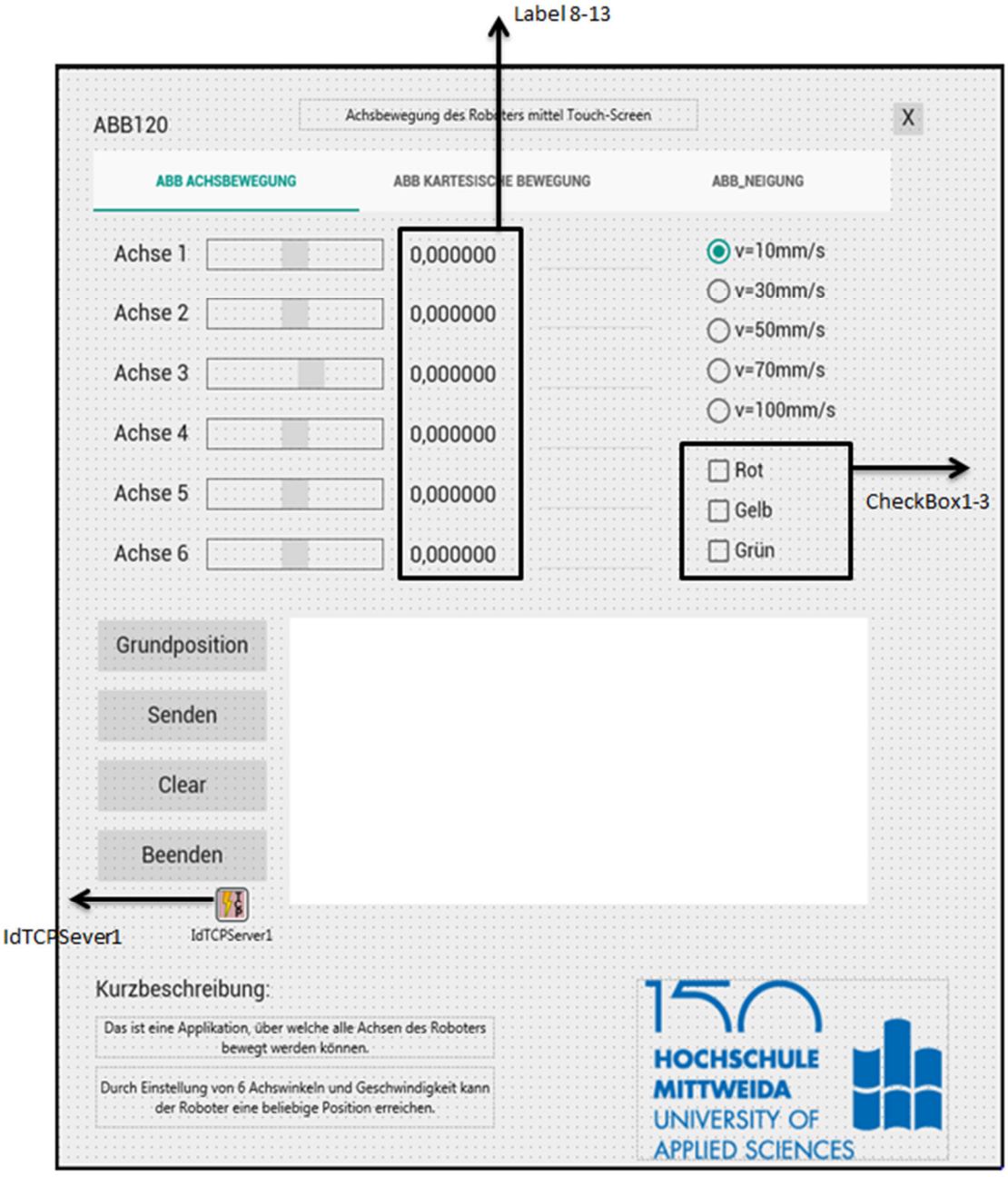

Abbildung 4.3-3: Das Bedienschnittstellendesign_1 auf der Delphi-Ansicht

4.3.2 Designideen

4.3.2.1 Server/Client-Kommunikation

Die Aufgabe verwendet der *IdTCPServer*-Komponente zur Realisierung der Client-Server-Kommunikation. Das Modul wird verwendet, um die Verbindung zwischen den zwei Maschinen zu erreichen, und unterstützt die folgenden Ereignisse: *OnConnect*, *OnDisconnect*, *OnExecute und OnException*.

Beim *OnConnect*-Ereignis nach der Verbindung wird die IP-Adresse in *Memo1* angezeigt und die rote Lampe und grüne Lampe werden aktiviert. Gleichzeitig wird *TabItem1* ausgewählt. Der Code ist wie folgt:

```
procedure TForm1.IdTCPServer1Connect(AContext: TIdContext);
begin
Memo1.Lines.Add('Verbindung: ' + AContext.Binding.PeerIP);
hContext := AContext;
CheckBox1.IsChecked:=True;
CheckBox3.IsChecked:=True;
TabItem1.IsSelected:=True;
end;
```

Beim *OnDisconnect*-Ereignis nach dem Verbindungsabbruch wird auch die IP-Adresse in *Memo1* zeigt. Der Code ist wie folgt:

```
procedure TForm1.IdTCPServer1Disconnect(AContext: TIdContext);
begin
Memo1.Lines.Add('Verbindungsabbruch: ' + AContext.Binding.PeerIP);
end;
```

Beim *OnException*-Ereignis zeigt in der Bedienschnittstelle das Wort Fehler, wenn die Lese- oder Schreibdaten nicht erfolgreich ausgeführt werden. Der Code ist wie folgt:

```
procedure TForm1.IdTCPServer1Exception(AContext: TIdContext;
AException: Exception);
begin
showmessage('Fehler')
end;
```

Beim *OnExcute*-Ereignis nach dem Empfangen von der Roboterantwort, die vom Server gesendet wurde, wird die Antwort in *Memo1* angezeigt. Der Code ist wie folgt:

```
procedure TForm1.IdTCPServer1Execute(AContext: TIdContext);
begin
Antwort:=AContext.Connection.IOHandler.ReadLn;
Memo1.Lines.Add('Nachricht: ' + Antwort);
end;
```

4.3.2.2 Die Eingabe der Achsewinkel

Jede Scrollbar entspricht einem Achswinkel, hat einen eigenen Wertbereich und einen eigenen Smallchange-wert. Wenn sich der Wert des Scrollbars ändert, zeigt das entsprechende Label den Echtzeitwert an. Deshalb wird das *OnChange*-Ereignis von Scrollbar verwendet. Der Code ist wie folgt:

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
Label8.Text := FloatToStrF(scrollbar1.Value, ffFixed, 8, 6);
end;
```

Scrollbar2, *Scrollbar3*, *Scrollbar4*, *Scrollbar5 und Scrollbar6* sind gleich. Bei der Achsbewegung muss maximal 6 Nachkommastellen verwendet werden. Deshalb wird die Funktion *FloatToStrF* mit Digits 6 verwendet.

Der Programmier kann auch einen beliebigen Wert im entsprechende Edit eingeben, um den Wert der entsprechenden Scrollbar flexibel festzulegen. Deshalb wird das *OnChange*-Ereignis von Edit verwendet. Der Code ist wie folgt:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
Scrollbar1.Value :=StrToFloat(Edit1.Text);
end;
```

Edit2, *Edit3*, *Edit4*, *Edit5 und Edit6* sind gleich.
Der entspechende Wertebereich von Scrollbar wird gemäß der folgenden Tabelle festgelegt:

Achse 1	Rotationsbewegung	+165° bis -165°
Achse 2	Armbewegung	+110° bis -110°
Achse 3	Armbewegung	+70° bis -110°
Achse 4	Handgelenkbewegung	+160° bis -160°
Achse 5	Neigebewegung	+120° bis -120°
Achse 6	Drehbewegung	+400° bis -400°

Tabelle 4.3-1: Type und Bereiche der Bewegung für jede Achse IRB120 (Quelle [14])

4.3.2.3 Die Einstellung der Geschwindigkeiten

Fünf RadioButtons entsprechen fünf Bewegungsgeschwindigkeiten v des Roboters. Deshalb wird das *OnClick*-Ereignis von *RadioButton1* verwendet. Der Code ist wie folgt:

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
gesch:= ' 10 ' ;
end;
```

RadioButton2, *RadioButton3*, *RadioButton4 und RadioButton5* sind gleich. Dann gibt es fünf Klassen $10\ mm/s, 30\ mm/s, 50\ mm/s, 70\ mm/s$ und $100\ mm/s$ in der Geschwindigkeit.

4.3.2.4 Die Einstellung der Lampen

Drei CheckBoxs entsprechen drei Lampenfarben des Roboters. Es kann ausgewählt oder nicht ausgewählt werden. Die Benutzer können klicken, um auszuwählen und nochmal klicken, um die Auswahl aufzuheben. Deshalb wird das *OnChange*-Ereignis von Check-Box verwendet. Der Code über die rote Lampe ist wie folgt:

```
procedure TForm1.CheckBox1Change(Sender: TObject);
begin
if CheckBox1.IsChecked=True then
hContext.Connection.IOHandler.WriteLn('set_digital_output(1,True)');
        // rote Lampe  anschalten
if CheckBox1.IsChecked=False then
hContext.Connection.IOHandler.WriteLn('set_digital_output(1,False)');
        // rote Lampe ausschalten
end;
```

CheckBox2 (gelbe Lampe) und *CheckBox3* (grüne Lampe) sind gleich.

4.3.2.5 String-Antwort zerschneiden

Die Antworten der Robotersteuerung liegen als String vor. Die sechs Achswinkel müssen extrahiert werden, um der entsprechende Wert des entsprechen Scrollbars anzuzeigen. (Siehe Anlagen „String-Antwort zerschneiden“)

4.3.2.6 Multifunktionsrealisierung

Die Schließung des Fensters:

Nach dem Drücken der Schließen-Button wird das gesamte Fenster geschlossen. Das *OnClick*-Ereignis von Schließen-Button wird verwendet.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
```

Die Sendung des Befehls:

Der Achswinkelswert muss durch den entsprechend Scrollbarwert im Bogenmaß ausgewechselt werden und gleichzeitig im Stringzeichen $txt1-6$ darstellen. $Txt1-6$ verwenden maximal 6 Nachkommastellen. Dann ersetzt StringReplace im $txt1-6$ das Komma durch den Punkt, damit die Robotersteuerung der Befehl erkennt. Anschließend wird das *OnClick*-Ereignis von Senden-Button verwendet. Der Code ist wie folgt:

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
txt1 := FloatToStrF(scrollbar1.Value*3.1415/180, ffFixed, 8, 6);
txt1:=StringReplace (txt1, ',', '.', []) ;
hContext.Connection.IOHandler.WriteLn
('movej(['+txt1+', '+txt2+', '+txt3+', '+txt4+', '+txt5+', '+txt6+'],v='+gesch+')');
end;
```

Am Ende sendet das Programm die 6 Achswinkel und Geschwindigkeit mit *movej*-Befehle, um der Roboter zu bewegen.

Beenden des Roboterprogramms:

Nach dem Drücken der Beenden-Button wird das Roboterprogramm beendet. Das *On-Click*-Ereignis von Beenden-Button wird verwendet. Der Exit-Befehl wird gesendet.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
hContext.Connection.IOHandler.WriteLn('exit');
end;
```

Die Löschung des Memoinhalts:

Nach dem Drücken der Clear-Button wird der Memoinhalt alles gelöscht. Das *OnClick*-Ereignis von Clear-Button wird verwendet.

```
procedure TForm1.Button4Click(Sender: TObject);
begin
Memo1.Lines.Clear;
end;
```

Bewegen in die Grundposition:

Nach dem Drücken der Grundposition-Button wird der Roboter in den Grundpunkt bewegt. Das *OnClick*-Ereignis von Grundposition-Button wird verwendet. Der Roboter bewegt in die Grundposition $0°,\ 0°,\ 0°,\ 0°,\ 90°,\ 0°$ nämlich kartesische Koordinate mit Eulerschen Winkel $[302,0\ ,558,-180,0,-180]$.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
hContext.Connection.IOHandler.WriteLn('movej([0,0,0,0,1.57,0],v=50)');
end;
```

4.3.3 Systemtest

Die Bedienschnittstelle ist klar, leicht zu verstehen und zu bedienen und kann Aufgaben direkt ausführen. Es ist notwenddig zu testen, ob die Kommunikation erfolgreich aufgebaut und abgebrochen wird, so werden die ersten und vierten Schritte ausgeführt. In der

Aufgabe muss der Programmier prüfen, ob der Befehl korrekt gesendet werden kann, so werden die zweiten und dritten Schritte ausgeführt.

① Herstellen eine Verbindung mit TCP/IP-Protokoll

② Einstellen der sechs Achswinkel auf $20°, -5°, 10°, -20°, 30°, -30°$ und empfangen der Roboterantwort nach dem Senden der Steueranweisungen mit Senden-Button

③ Drücken den Grundposition-Button und empfangen der Roboterantwort

④ Drücken den Beenden-Button und abbrechen der Verbindung

Der endgültige Operationseffekt ist wie folgt:

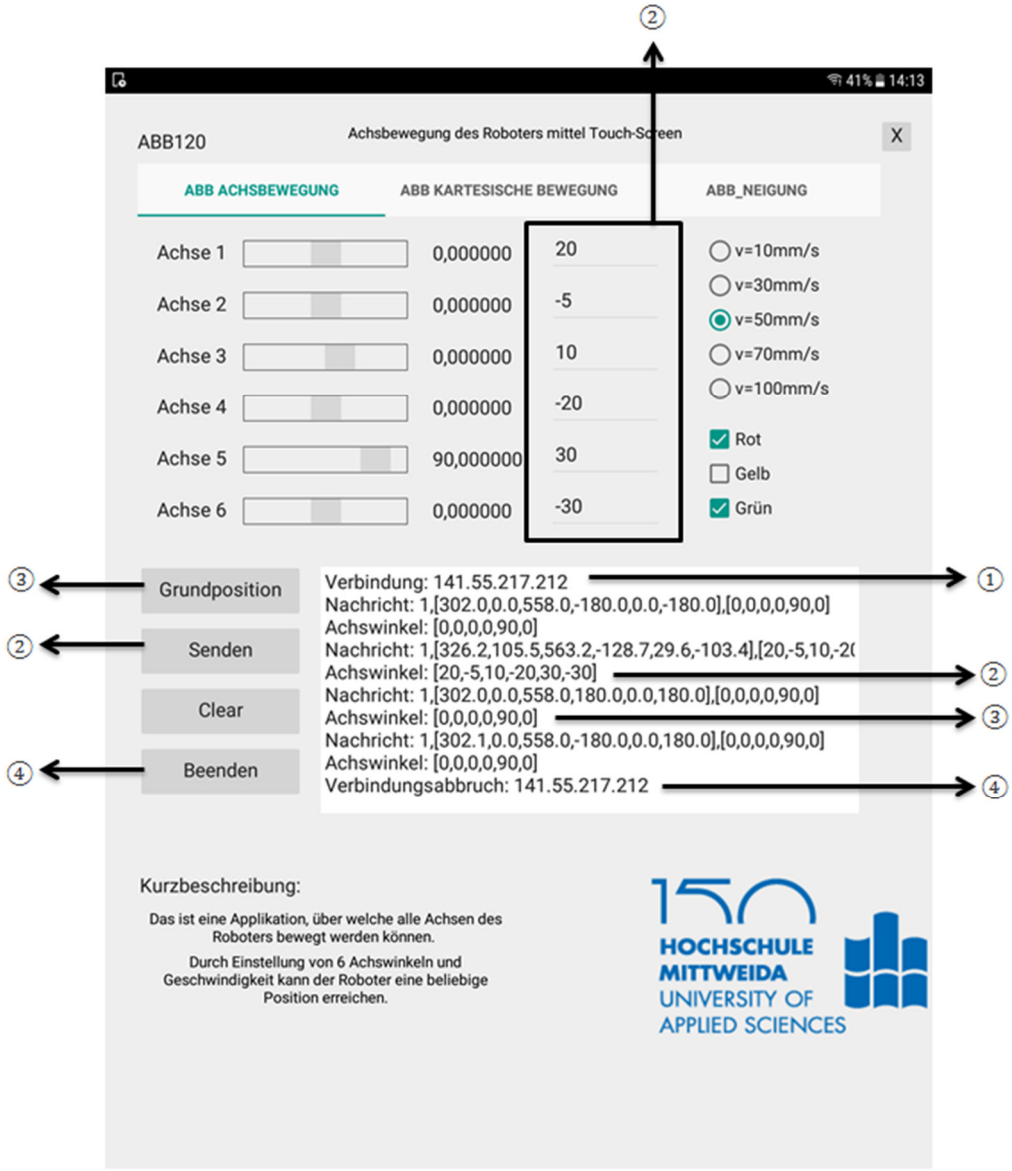

Abbildung 4.3-4: Der Systemtest der Teilaufgabe 1(Tablet)

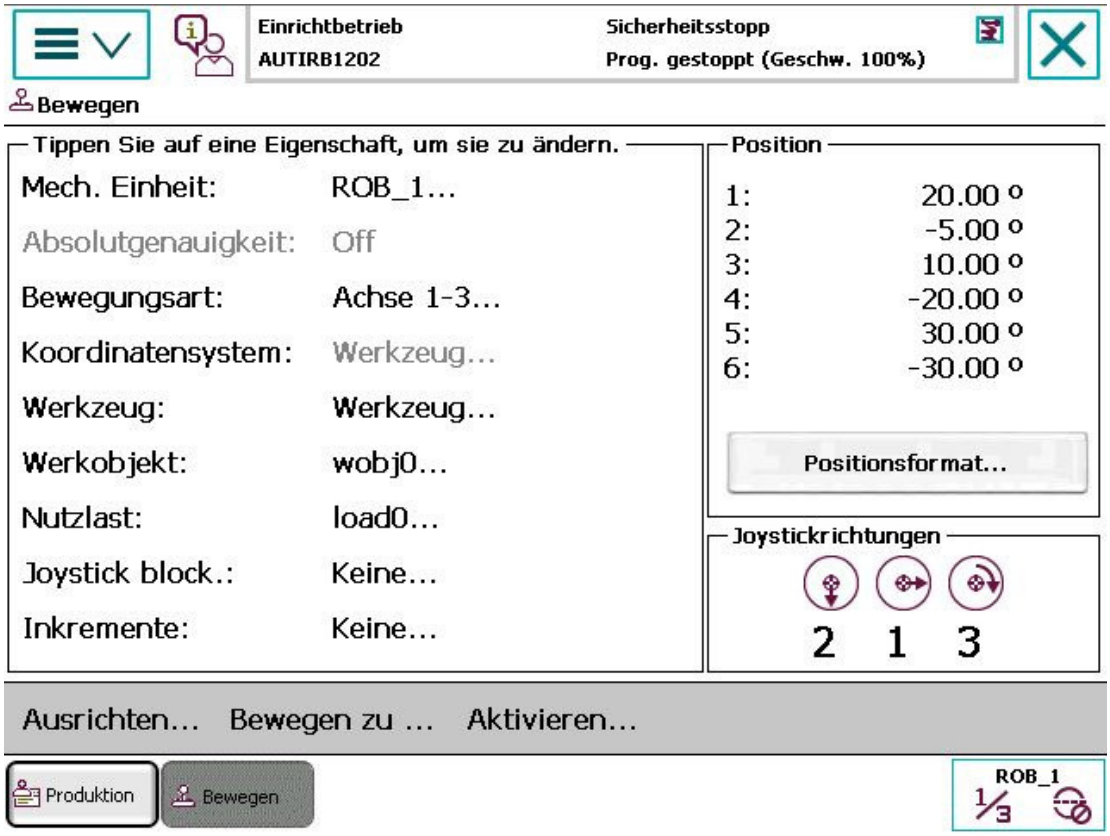

Abbildung 4.3-5: Achswinkelanzeige der Robotersteuerung

Testauswertung:

Die Kommunikation wird erfolgreich aufgebaut und abgebrochen. Die realen Achswinkel sind mit den vorgegebenen Winkeln $20°, -5°, 10°, -20°, 30°, -30°$ identisch. Nach dem Drücken der Grundposition-Button wird der Roboter wie der Bediener will in die Grundposition $0°,\ 0°,\ 0°,\ 0°,\ 90°,\ 0°$ bewegt.

4.4 Kartesische Bewegung des Roboters mittels Touch-Screen

Das ist eine Applikation, welche es ermöglicht, den Roboter im Welt-Koordinatensystem zu bewegen. Mithilfe der 6-Richtungs-Button, der Verschiebungs-Distanz und der Geschwindigkeit kann der Roboter im Welt-Koordinatensystem kartesische Bewegungen ausführen.

4.4.1 Bedienschnittstellendesign

Das Bedienschnittstellendesign der zweiten Teilaufgabe auf der Android-Ansicht ist in der Abbildung 4.4-1 dargestellt:

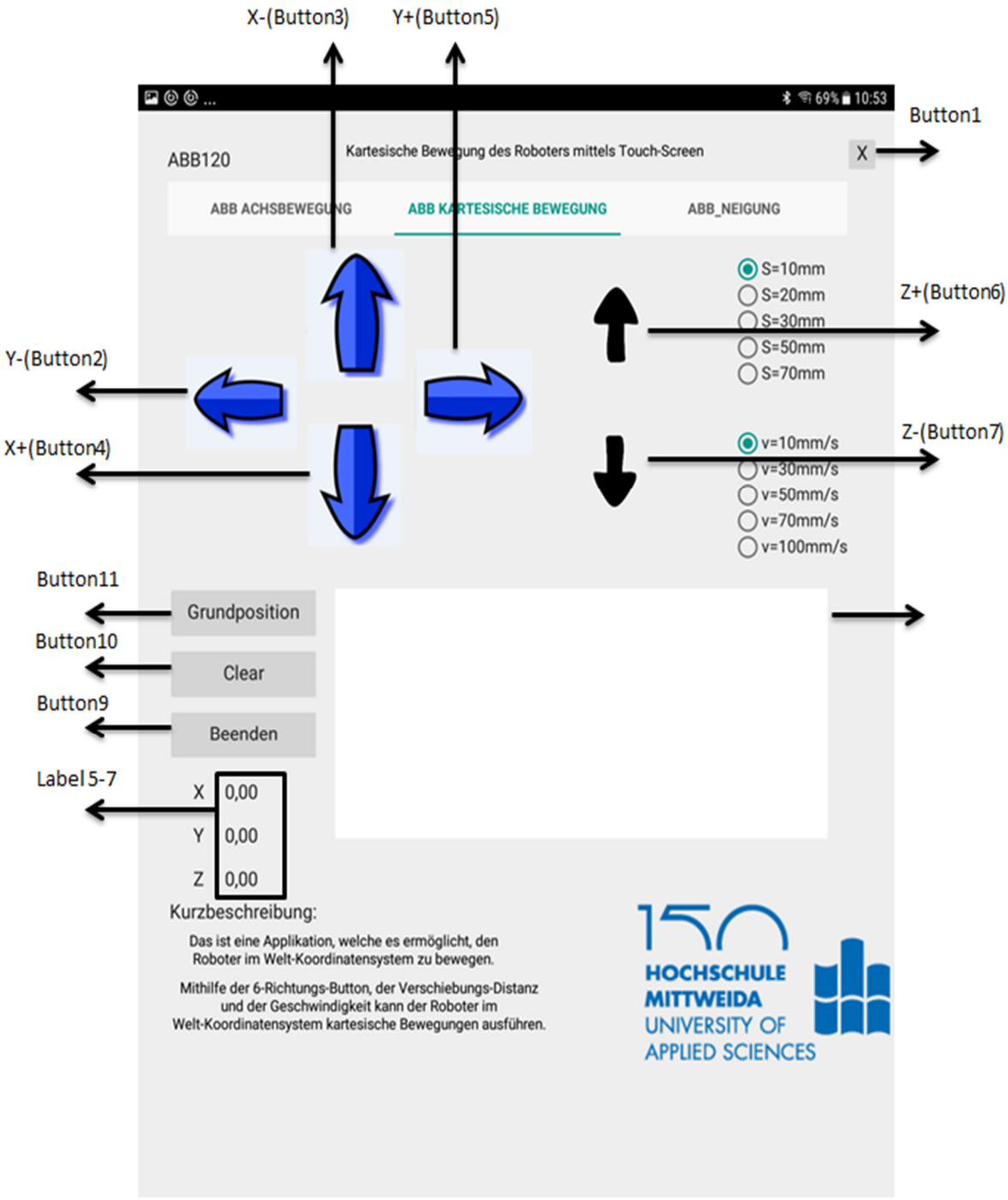

Abbildung 4.4-1: Das Bedienschnittstellendesign_2 auf der Android-Ansicht

Das Bedienschnittstellendesign der zweiten Teilaufgabe auf der Delphi-Ansicht ist in der Abbildung 4.4-2 dargestellt:

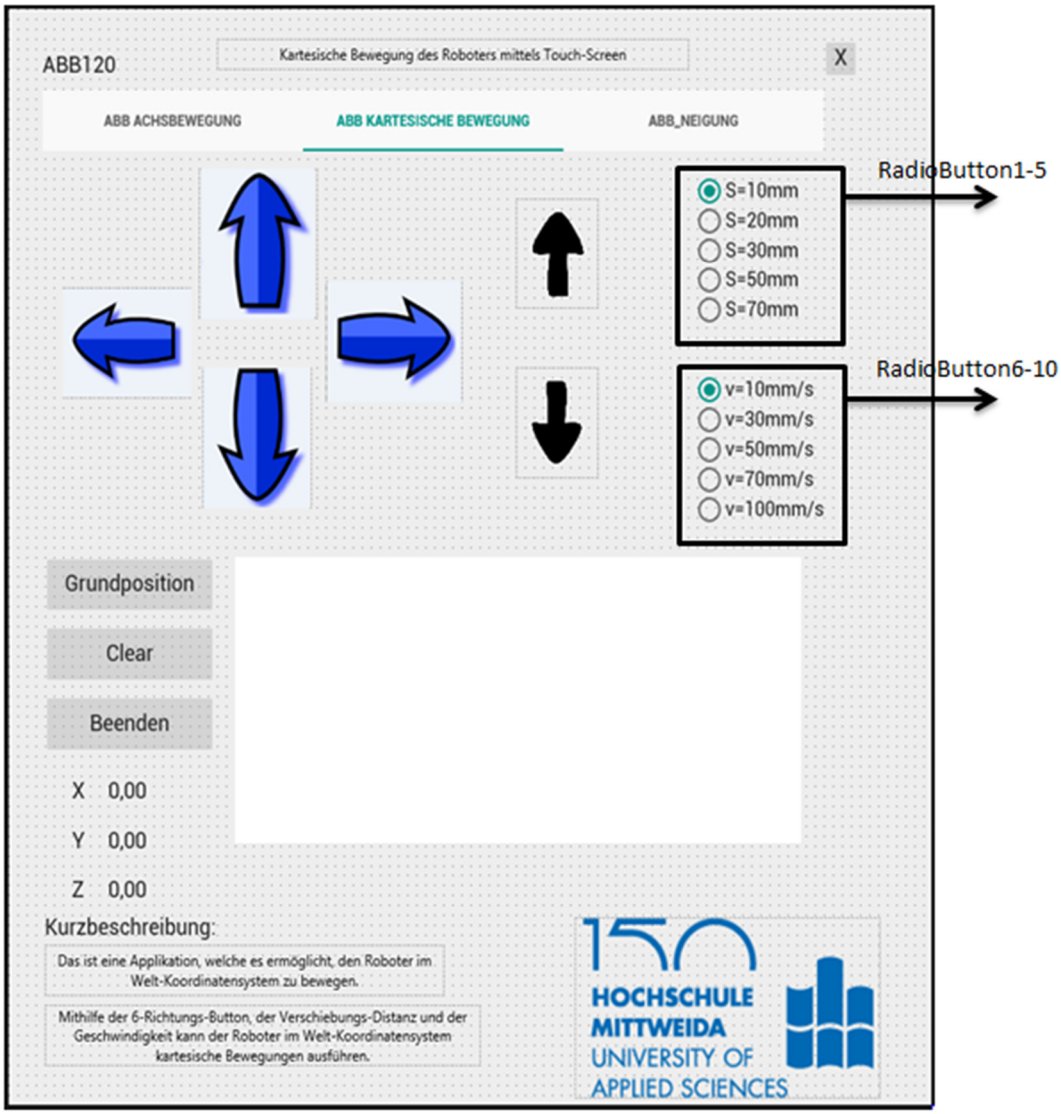

Abbildung 4.4-2: Das Bedienschnittstellendesign_2 auf der Delphi-Ansicht

4.4.2 Designideen

Form1 hat bereits eine Verbindung mit der Robotersteuerung hergestellen, einige Befehle von Form2 müssen nur direkt den *IdTCPServer* von Form1 kommunizieren. Die Kommunikation wird folgendermaßen aufgebaut:

```
Form1.hContext.Connection.IOHandler.WriteLn
('movej([0,0,0,0,1.57,0],v=50)');
```

Der Benutzer fügt eine Deklaration in die Uses-Anweisung im Abschnitt Implementierung ein. Wie folgt:

```
implementation
uses ABBachsbewegung;
```

4.4.2.1 Die Sechs-Richtung-Tasteneinstellungen

Jeder Richtungsbutton entspricht einer Verschiebung nach einer entsprechenden Richtung. Die Verschiebung hat einen eigenen Grenzwert. Und jeder Richtungsbutton hat eigenes entsprechendes Tastebild. Deshalb wird das *OnClick*-Ereignis von Button2-7 verwendet. Der Code über die X+Richtung ist wie folgt (String-*eins* ist die X-Position und String-*S* ist die Verschiebungsdistanz):

```
procedure TForm2.Button4Click(Sender: TObject);
begin
eins:=StringReplace (eins, '.', ',', []) ;
        // StringReplace in der X+Position der Punkt durch das Komma ersetzen, um die
        StrToFloat-Funktion zu erkennen
        Ein Beispiel:
        Das Rückgabeergebnis von String-eins '302.2'  ist '302,2'.
f_eins=StrToFloat(eins);
if (f_eins<400) and (f_eins +StrToFloat(S)<=400) then
Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_actual_tcp_pose(),['+S+',0,
0,0,0,0]),v='+gesch2+')')
        // Nach der Verschiebung wird die aktuelle Position_X+ auch innerhalb 400 mm, dann be-
        wegt der Roboter in X+ in S mm
else if (f_eins <400) and (f_eins +StrToFloat(S)>400) then
Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_actual_tcp_pose(),['+FloatT
oStr(400-StrToFloat(eins))+',0,0,0,0,0]),v='+gesch2+')');
end;
        // Nach der Verschiebung wird die aktuelle Position_X+ mehr als 400mm, dann bewegt der
        Roboter in X+ direkt bis Maximalwert
```

Der Maximalwert von X+Richtung ist $400\ mm$. String-$eins$ ist die X-Position von der Roboterantwort. Dann ersetzt StringReplace im $eins$ der Punkt durch das Komma, um die StrToFloat-Funktion zu erkennen.

Der Minimalwert von X-Richtung ist $100\ mm$. Der Maximalwert von Y+Richtung ist $250\ mm$. Der Minimalwert von Y-Richtung ist $-250\ mm$. Der Maximalwert von Z+Richtung ist $600\ mm$. Der Minimalwert von Z-Richtung ist $200\ mm$. Für diese fünf Fälle sind gleiche Vorgehensweise.

4.4.2.2 Die Einstellung der Verschiebungsdistanz (Gruppe 1)

Fünf RadioButtons entsprechen fünf kartesische Bewegungsdistanz s des Roboters in xyz-Richtung im Welt-Koordinatensystem. Die Optionen in einer Gruppe von Optionsfeldern schließen sich gegenseitig aus, wobei immer nur ein Optionsfeld ausgewählt werden

kann. Deshalb wird das *OnClick*-Ereignis von *RadioButton1* verwendet. Der Code ist wie folgt:

```
procedure TForm2.RadioButton1Click(Sender: TObject);
begin
S:= ' 10 ' ;
end;
```

RadioButton2, *RadioButton3*, *RadioButton4 und RadioButton5* sind gleich. Dann gibt es fünf Klassen $10\ mm,\ 20\ mm,\ 30\ mm,\ 50\ mm$ und $70\ mm$ in der Verschiebungsdistanz.

4.4.2.3 Die Einstellung der Geschwindigkeiten (Gruppe 2)

Fünf RadioButtons entsprechen fünf Bewegungsgeschwindigkeiten v des Roboters. Die Optionen in einer Gruppe von Optionsfeldern schließen sich gegenseitig aus, wobei immer nur ein Optionsfeld ausgewählt werden kann. Deshalb wird das *OnClick*-Ereignis von *RadioButton6* verwendet. Der Code ist wie folgt:

```
procedure TForm2.RadioButton6Click(Sender: TObject);
begin
gesch2:= ' 10 ' ;
end;
```

RadioButton7, *RadioButton8*, *RadioButton9 und RadioButton10* sind gleich. Dann gibt es fünf Klassen $10\ mm/s,\ 30\ mm/s,\ 50\ mm/s,\ 70\ mm/s$ und $100\ mm/s$ in der Geschwindigkeit.

4.4.2.4 String-Antwort zerschneiden und anzeigen

Die Antworten der Robotersteuerung liegen als String vor. Die drei Positionsdaten müssen extrahiert werden, um der entsprechende Wert des entsprechen Labels anzuzeigen. (Siehe Anlagen „String-Antwort zerschneiden“)

String-Antwort anzeigen:

```
Label5.Text := eins;          // X – Position anzeigen
Label6.Text := zwei;          // Y – Position anzeigen
Label7.Text := drei;          // Z – Position anzeigen
```

4.4.2.5 Multifunktionsrealisierung

Die Schließung des Fensters:

Nach dem Drücken der Schließen-Button wird das gesamte Fenster geschlossen. Das *OnClick*-Ereignis von Schließen-Button wird verwendet.

```
procedure TForm2.Button1Click(Sender: TObject);
```

```
begin
Form1.Close;
end;
```

Beenden des Roboterprogramms:

Nach dem Drücken der Beenden-Button wird der Roboterverlauf beendet. Das *OnClick*-Ereignis von Beenden-Button wird verwendet. Der Exit-Befehl wird gesendet.

```
procedure TForm2.Button9Click(Sender: TObject);
begin
Form1.hContext.Connection.IOHandler.WriteLn('exit');
end;
```

Die Löschung des Memoinhalts:

Nach dem Drücken der Clear-Button wird der Memoinhalt alles gelöscht. Das *OnClick*-Ereignis von Clear-Button wird verwendet.

```
procedure TForm2.Button10Click(Sender: TObject);
begin
Memo1.Lines.Clear;
end;
```

Bewegen in die Grundposition:

Nach dem Drücken der Grundposition-Button wird der Roboter in den Grundpunkt bewegt. Das *OnClick*-Ereignis von Grundposition-Button wird verwendet. Der Roboter bewegt in die Grundposition $0°,\ 0°,\ 0°,\ 0°,\ 90°,\ 0°$ nämlich kartesische Koordinate mit Eulerschen Winkel $[302,0\ ,558,-180,0,-180]$.

```
procedure TForm2.Button11Click(Sender: TObject);
begin
Form1.hContext.Connection.IOHandler.WriteLn('movej([0,0,0,0,1.57,0],v=50)');
end;
```

4.4.3 Systemtest

Die Bedienschnittstelle ist klar, leicht zu verstehen und zu bedienen und kann Aufgaben direkt ausführen.

① Herstellen eine Verbindung mit TCP/IP-Protokoll (Siehe Abb. 4.3-3)

② Befehltest:

a. Drücken den X+Button mit $50\ mm$-Verschiebung und $30\ mm/s$-Geschwindigkeit und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

b. Drücken den Y-Button mit $70\,mm$-Verschiebung und $30\,mm/s$-Geschwindigkeit und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

c. Drücken den Z+Button mit $30\,mm$-Verschiebung und $50\,mm/s$-Geschwindigkeit und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

Der endgültige Operationseffekt ist wie folgt:

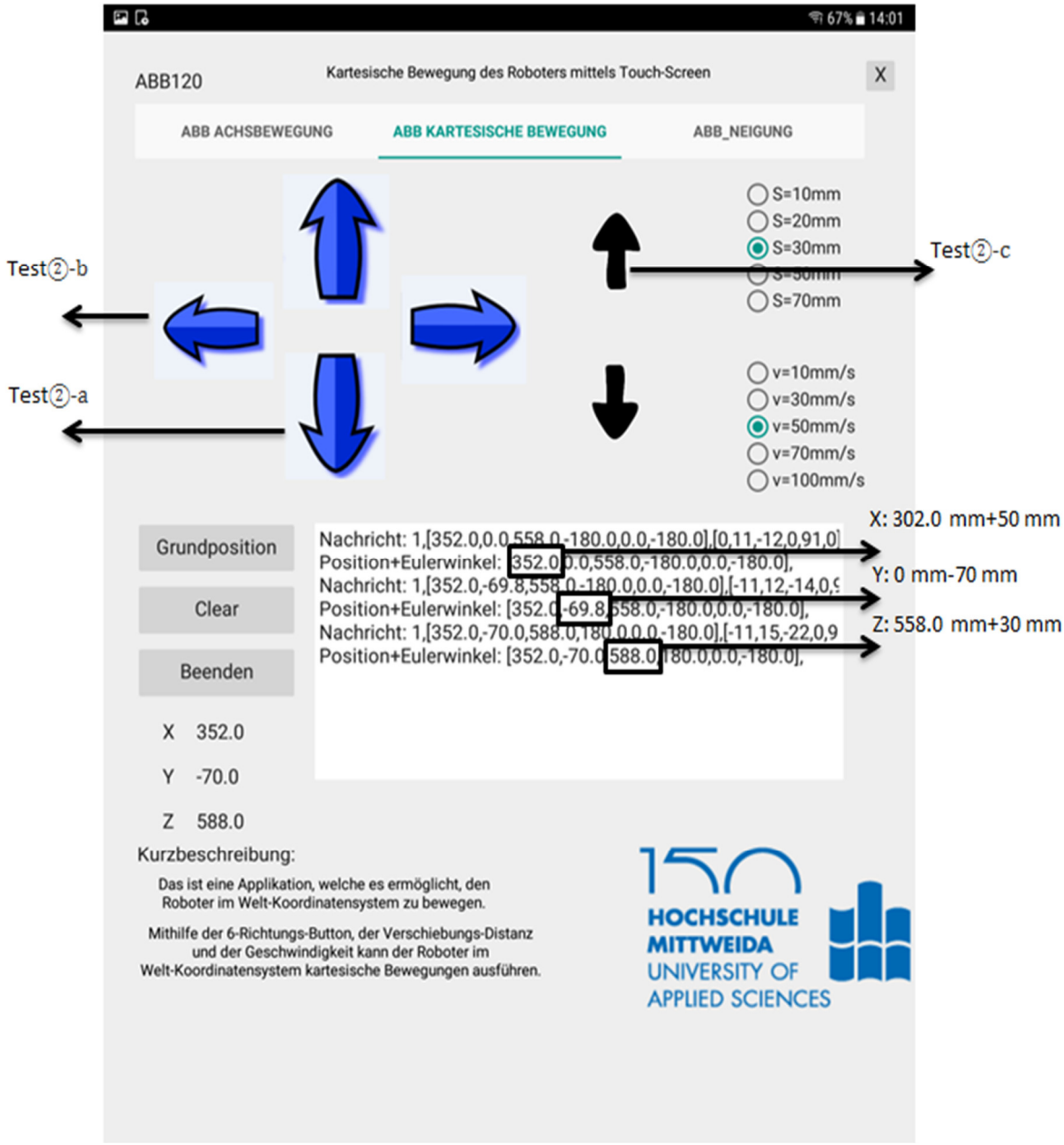

Abbildung 4.4-3: Das Befehltest der Teilaufgabe 2(Tablet)

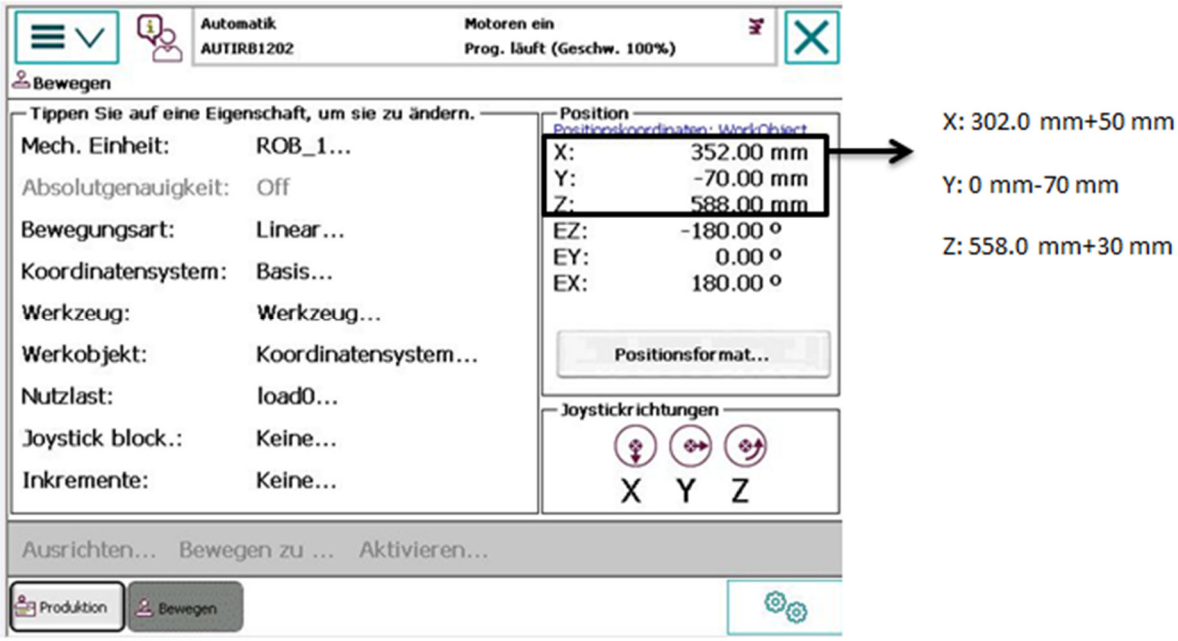

Abbildung 4.4-4: Das Befehltest der Teilaufgabe 2(Roboter)

③Grenzwerttest:

a. Drücken den X+Button mit $50\,mm$-Verschiebung und $30\,mm/s$-Geschwindigkeit bis $400\,mm$ und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

b. Drücken den Y+Button mit $70\,mm$-Verschiebung und $30\,mm/s$-Geschwindigkeit bis $250\,mm$ und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

c. Drücken den Z+Button mit $30\,mm$-Verschiebung und $50\,mm/s$-Geschwindigkeit bis $600\,mm$ und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

Der endgültige Operationseffekt ist wie folgt:

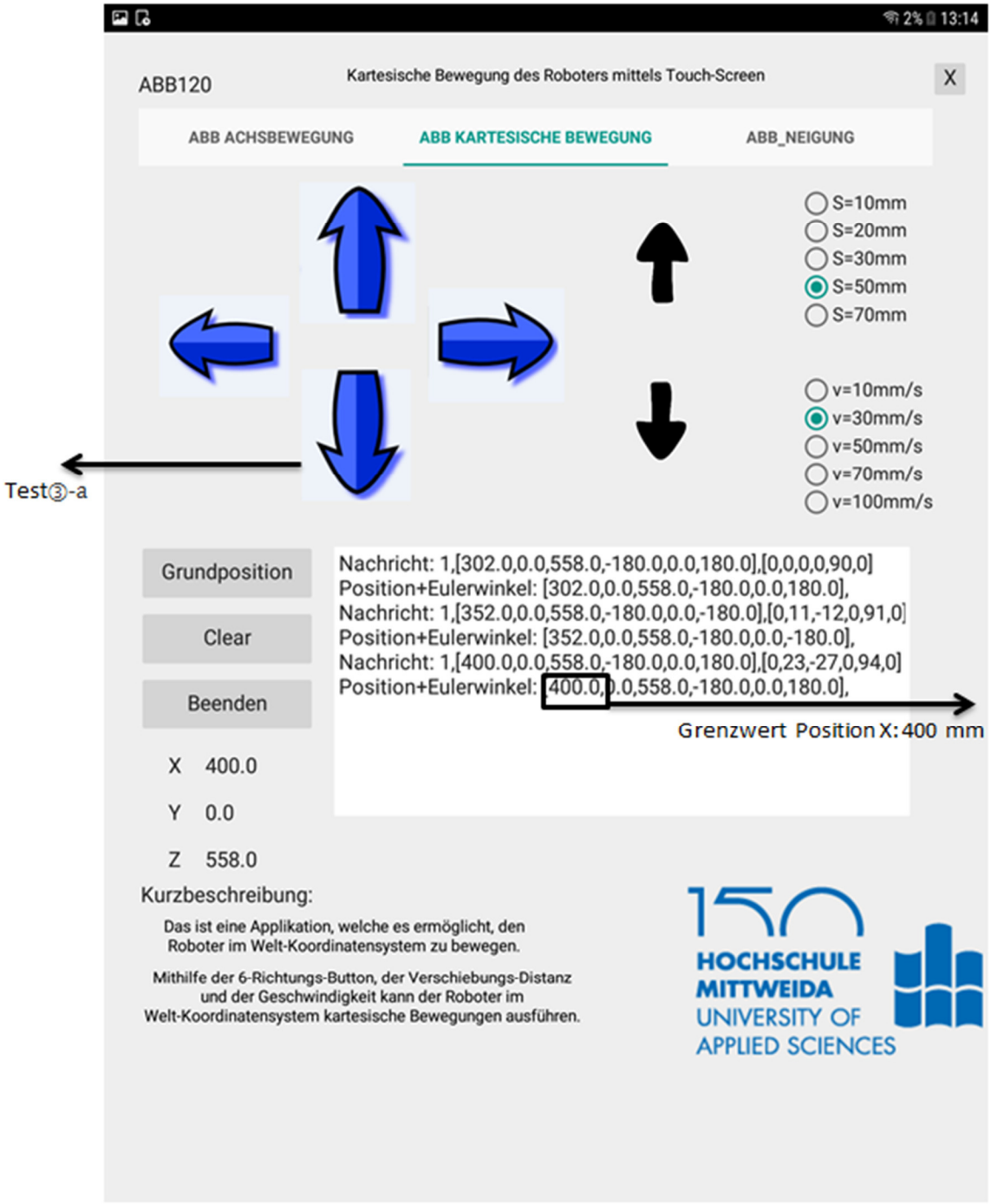

Abbildung 4.4-5: Das Grenzwerttest_PositionX der Teilaufgabe 2(Tablet)

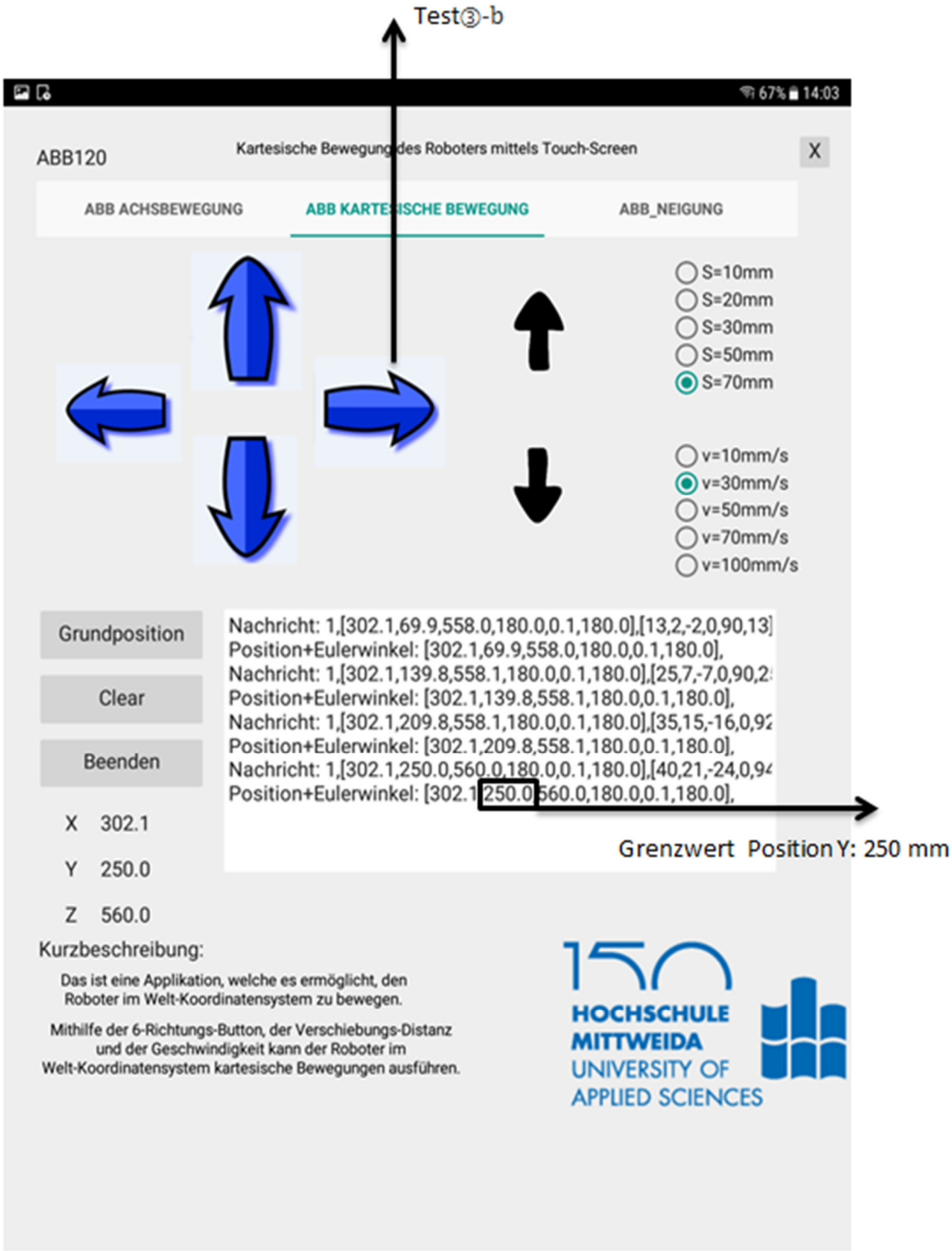

Abbildung 4.4-6: Das Grenzwerttest_PositionY der Teilaufgabe 2(Tablet)

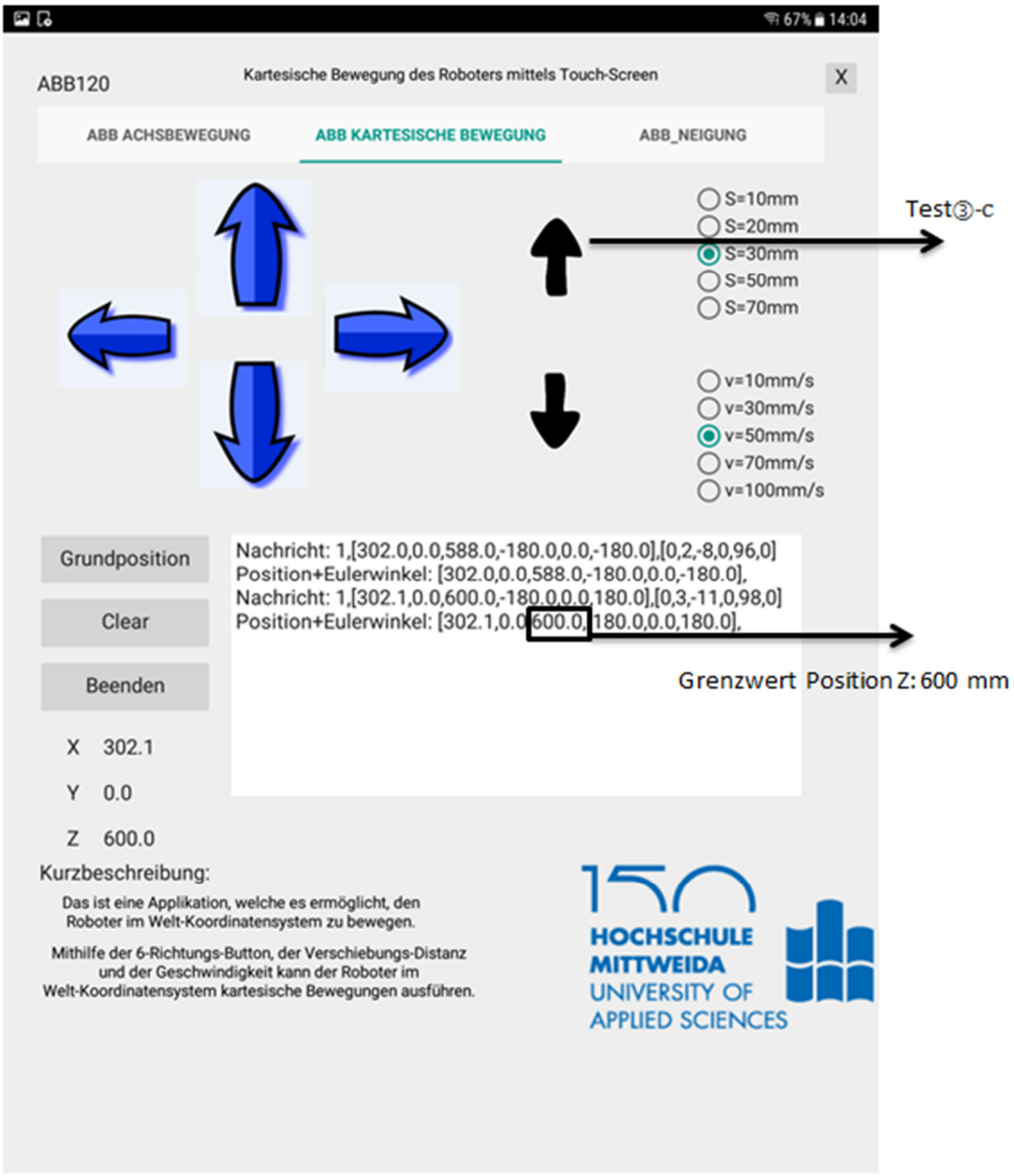

Abbildung 4.4-7: Das Grenzwerttest_PositionZ der Teilaufgabe 2(Tablet)

④ Drücken den Grundposition-Button und empfangen die XYZ-Positon von der Roboter-antwort

Der endgültige Operationseffekt ist wie folgt:

Abbildung 4.4-8: Der Grundpositiontest der Teilaufgabe 2(Roboter)

⑤ Drücken den Beenden-Button und abbrechen die Verbindung (Siehe Abb. 4.3-3)

Testauswertung:

Die Kommunikation wird erfolgreich aufgebaut und abgebaut. Die reale Verschiebungen in drei Richtungen sind mit den vorgegebenen Wert X+ $50\,mm$, Y+ $70\,mm$, Z+ $30\,mm$ identisch und können korrekt anzeigen. Der Grenzwert von XYZ entsprechen dem Bereich. Nach dem Drücken der Grundposition-Button wird der Roboter wie der Bediener will in die Grundposition $0°,\ 0°,\ 0°,\ 0°,\ 90°,\ 0°$ nämlich kartesische Koordinate mit Eulerschen Winkel $[302,0\,,558,-180,0,-180]$ bewegt.

4.5 Kartesische Bewegung des Roboters mittels Neigungs-Sensor

Das Neigen des Tablets bewirkt eine lineare Bewegung des Roboters.

Je größer der Neigungswinkel ist, desto weiter ist die lineare Verschiebung.

4.5.1 Bedienschnittstellendesign

Das Bedienschnittstellendesign der dritten Teilaufgabe auf der Android-Ansicht ist in der Abbildung 4.5-1 dargestellt:

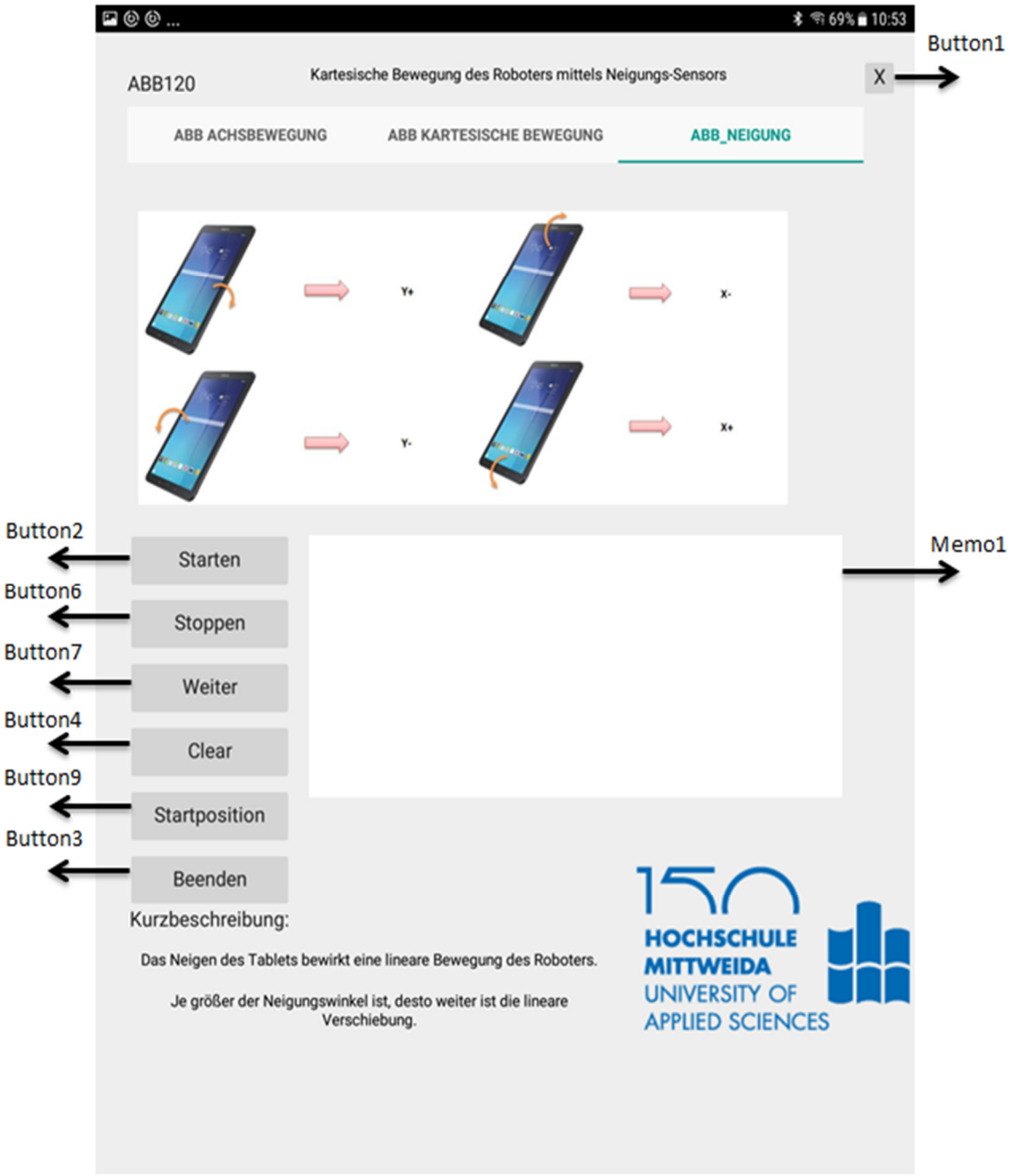

Abbildung 4.5-1: Das Bedienschnittstellendesign_3 auf der Android-Ansicht

Das Bedienschnittstellendesign der dritten Teilaufgabe auf der Delphi-Ansicht ist in der Abbildung 4.5-2 dargestellt:

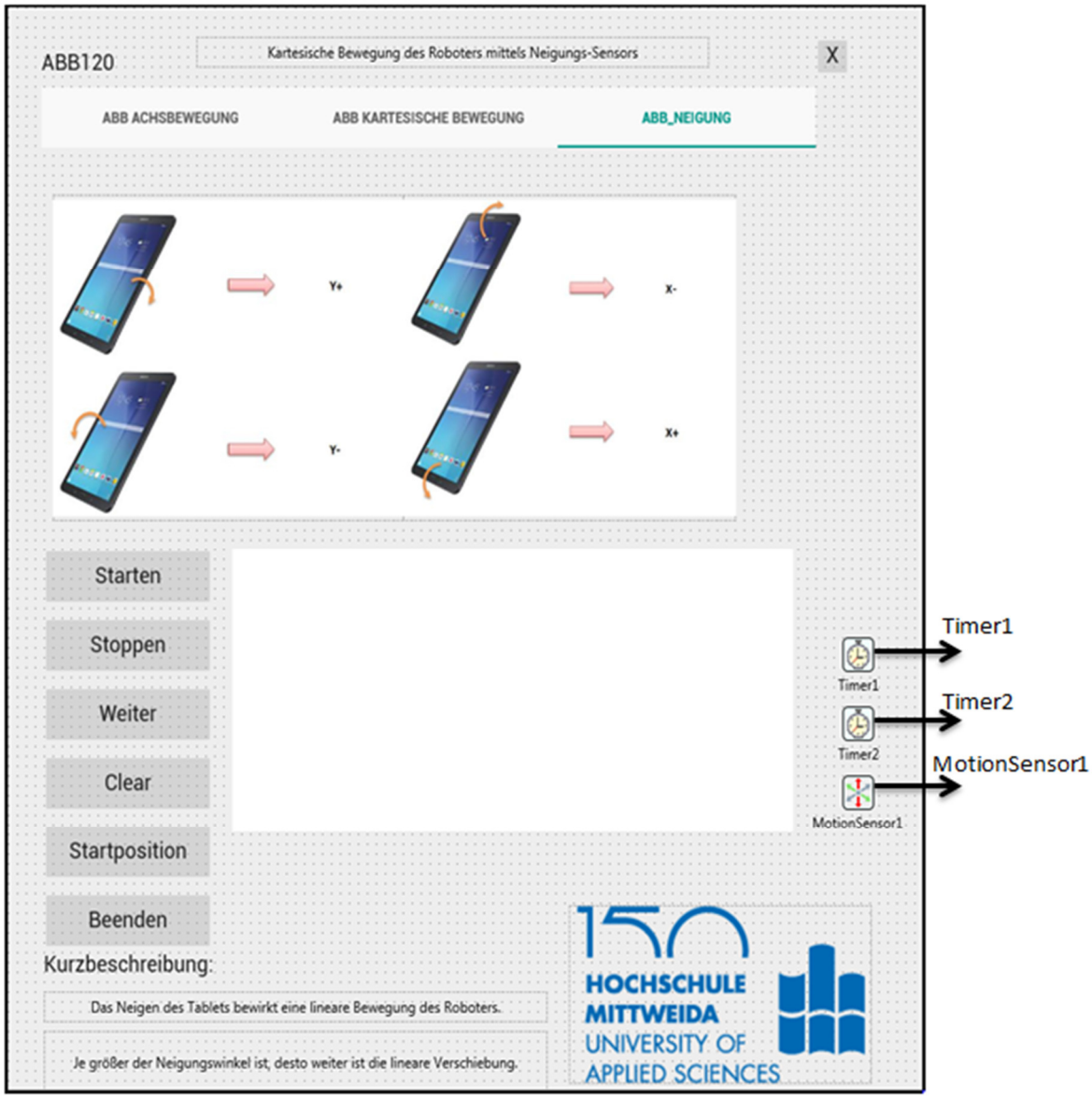

Abbildung 4.5-2: Das Bedienschnittstellendesign_3 auf der Delphi-Ansicht

4.5.2 Designideen

Ebenso wie Kapitel 4.4.2 hat Form1 bereits eine Verbindung mit der Robotersteuerung hergestellen, einige Befehle von Form3 müssen nur direkt den *IdTCPServer* von Form1 kommunizieren.

Der Benutzer fügt auch eine Deklaration in die Uses-Anweisung im Abschnitt Implementierung ein. Wie folgt:

```
implementation
uses ABBachsbewegung;
```

4.5.2.1 String-Antwort zerschneiden und anzeigen

Der Parameter des MotionSensors und die Antworten der Robotersteuerung liegen als String vor. Die drei Positionsdaten müssen extrahiert werden, um die entsprechende Position zu beurteilen. String-$zwei$ (Sensor_Y+ =Roboter_X-) repräsentiert hier PositionX und String-$eins$ (Sensor_X+ =Roboter_Y+) repräsentiert hier PositionY.
(Siehe Anlagen „String-Antwort zerschneiden“)

4.5.2.2 Standardeinstellungen und Neigungseinstellungen

Der Roboter_VerschiebungX Grenzwert: $100\ mm$ bis $400\ mm$

Der Roboter_VerschiebungY Grenzwert: $-250\ mm$ bis $250\ mm$

Der Tablet_Neigungsstartwert: 27° (Betrag) in jeder Richtung

Neigungsstartwert 27°: Wenn der Neigungswinkel innerhalb von $-27°$ bis 27° liegt, ist er nicht geneigt, d.h. er ist immer noch horizontal und tut nichts. Wenn der Neigungswinkel größer als 27° ist, bewegt sich danach der Roboter.

Der Wertbereich von *AccelerationX*(Label3), *AccelerationY*(Label2): $0\ cm/s^2$ - $1\ cm/s^2$

Deshalb $0.5 * 1000 * StrToFloat(Label2/3.text)$:
$0 – 500$ (ohne Einheit) entsprechen Neigungswinkel $0°$ - $90°$ (dann 150 entsprechen 27°)
Die Neigungsstartwerte oben wurden empirisch ermittelt.

Die Schritt-für-Schritt Analyse über Roboterachse_Y ist wie folgt:

```
eins:=StringReplace (eins, '.', ',', []) ;
        // StringReplace in der Y-Position der Punkt durch das Komma ersetzen, um die
        StrToFloat-Funktion zu erkennen.
if (0.5*1000*StrToFloat(Label2.text)<150) and (0.5*1000*StrToFloat(Label2.text)>-150) then
eins2:=FloatToStr(0)
        // Wenn der Neigungswinkel auf der Y-Achse innerhalb von -27° bis 27° liegt, ist er nicht
        geneigt, d.h. er ist immer noch horizontal und tut nichts.
else if (0.5*1000*StrToFloat(Label2.text)>150) and (StrToFloat(eins)<250) then
eins2:=FloatToStr(0.1*0.5*1000*StrToFloat(Label2.text))
        // Wenn der Neigungswinkel auf der Y-Achse größer als 27 ° ist und die Robo-
        ter_VerschiebungY weniger als 250mm beträgt, bewegt sich der Roboter gemäß dem Mo-
        tionSensordatum in Echtzeit.
else if (0.5*1000*StrToFloat(Label2.text)>150) and (StrToFloat(eins)>250) then
eins2:=FloatToStr(0)
        // Wenn der Neigungswinkel auf der Y-Achse größer als 27 ° ist und die Robo-
        ter_VerschiebungY größer als 250mm beträgt, bewegt sich der Robote in Y-Richtung nicht.
else if (0.5*1000*StrToFloat(Label2.text)<-150) and (StrToFloat(eins)>-250) then
eins2:=FloatToStr(0.1*0.5*1000*StrToFloat(Label2.text))
```

```
        // Wenn der Neigungswinkel auf der Y-Achse kleiner als -27 ° ist und die Roboter_VerschiebungY größer als -250mm beträgt, bewegt sich der Roboter gemäß dem MotionSensordatum in Echtzeit.
else if (0.5*1000*StrToFloat(Label2.text)<-150) and (StrToFloat(eins)<-250) then
eins2:=FloatToStr(0);
        // Wenn der Neigungswinkel auf der Y-Achse kleiner als -27 ° ist und die Roboter_VerschiebungY kleiner als -250mm beträgt, bewegt sich der Roboter in Y-Richtung nicht.
```

Die Schritt-für-Schritt Analyse über Roboterachse_X ist wie folgt:

Da diese Berechnung gleich Roboterachse_Y ist, dann wird alles in Anlage_ Programmierung_Form3 dargestellt.

Die Schritt-für-Schritt Diskussion über Grenzfall:

```
if ((StrToFloat(eins)+0.1*0.5*1000*StrToFloat(Label2.text))>250) then
eins2:=FloatToStrF(250-StrToFloat(eins), ffFixed, 8, 2)
else if (StrToFloat(eins)+0.1*0.5*1000*StrToFloat(Label2.text))<-250 then
eins2:=FloatToStrF(-250-StrToFloat(eins), ffFixed, 8, 2);
        // Wenn die Y-Achse nach der Verschiebung den Grenzwert überschreitet, bewegt sich die Y-Achse schließlich nur noch zur Grenze.

if (StrToFloat(zwei)-0.1*0.5*1000*StrToFloat(Label3.text))>400 then
 zwei2:=FloatToStrF(400-StrToFloat(zwei), ffFixed, 8, 2)
else if (StrToFloat(zwei)-0.1*0.5*1000*StrToFloat(Label3.text))<100 then
zwei2:=FloatToStrF(100-StrToFloat(zwei), ffFixed, 8, 2);
        // Wenn die X-Achse nach der Verschiebung den Grenzwert überschreitet, bewegt sich die Y-Achse schließlich nur noch zur Grenze.
```

Die Sendung des Verschiebungsbefehls:

Der Bewegungsbefehl wird gesendet, wenn mindestens eine Verschiebung in den zwei Achsenrichtungen vorliegt, um zu vermeiden, dass der Roboter $0\,mm$ -Verschiebungsbefehl wiederholend empfängt. Der Code ist wie folgt:

```
if (StrToFloat(zwei2)<>0) or (StrToFloat(eins2)<>0) then
begin
eins2:=StringReplace (eins2, ',', '.', []);
        // StringReplace in der Y-Position das Komma durch den Punkt ersetzen, um der Roboter den Befehl zu erkennen.
zwei2:=StringReplace (zwei2, ',', '.', []);
        // StringReplace in der X-Position das Komma durch den Punkt ersetzen, um der Roboter den Befehl zu erkennen.
Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_actual_tcp_pose(),['+zwei2
+','+eins2+',0,0,0,0]),v=30)');
end ;
```

Wiederneigung nach der Platzierung horizontal:

Nach der Neigung des Tablets bewegt sich der Roboter kontinuierlich. Wenn das Tablet wieder horizontal wird, stoppt der Roboter seine Bewegung. Wenn das Tablet zu diesem Zeitpunkt weiter kippt, führt der Roboter weiterhinentsprechende Bewegungen aus. Alle dies geschieht innerhalb von *Button8Click*-Ereignis. Die Eigenschaft Button8_Visible ist *False* eingestellt, weil *Button8* automatisch im *OnTimer*-Ereignis von *Timer1* gedrückt wird.

```
procedure TForm3.Button8Click(Sender: TObject);
begin
if (0.5*1000*StrToFloat(Label2.text)>150)
or (0.5*1000*StrToFloat(Label2.text)<-150)
or (-0.5*1000*StrToFloat(Label3.text)>150)
or (-0.5*1000*StrToFloat(Label3.text)<-150)  then
begin
if Runde=0 then
Button5Click(self);
Runde:=1;
end;
```

// Der Standardwert von Integer ' Runde ' ist 0. Wenn die Neigung der Tablet in vier Richtungen um mindestens eins größer ist als der Neigungsstartwert, wird Button5Click- Methoden aufgerufen und der Wert von Runde auf 1 geändert. Der Inhalt in Button5Click-Methoden ist konkrete Neigungseinstellungen.

```
if Runde=1 then
begin
if   (-0.5*1000*StrToFloat(Label3.text)<150)
and (-0.5*1000*StrToFloat(Label3.text)>-150)
and (0.5*1000*StrToFloat(Label2.text)<150)
and (0.5*1000*StrToFloat(Label2.text)>-150) then
Runde:=0;
end;
```

// Nachdem der Wert von Runde auf 1 geändert wird, wenn das Tablet wieder horizontal wird, wird der Wert von Runde auf 0 zurückgesetzt.

4.5.2.3 Implementierung der Zykluserkennung

Die Zykluserkennung wird durch das Zusammenspiel zweier Zähler realisiert.

Die Zeitdauer von *Timer1* wird auf eine Millisekunde eingestellt, um dem Label Echtzeitwerte im *OnTimer*-Ereignis zuzuweisen und *Button8Click*-Methoden ständig aufzurufen. Der Inhalt in *Button8Click*-Methoden ist die Erstneigungseinstellung und Wiederneigungseinstellung.

```
procedure TForm3.Timer1Timer(Sender: TObject);
begin
Label2.Text := FloatToStrF(Motionsensor1.Sensor.AccelerationX,ffFixed, 8, 2);
        // AccelerationX zeigen
Label3.Text := FloatToStrF(Motionsensor1.Sensor.AccelerationY,ffFixed, 8, 2);
        // AccelerationY zeigen
Button8Click(self);
end;
```

Die Zeitdauer von *Timer2* wird auf 100 Millisekunden eingestellt, weil *Button5Click*-Methoden nur rechtzeitig aufgerufen werden kann. Der Inhalt in *Button5Click*-Methoden ist konkrete Neigungseinstellungen. Der Start von *Timer2* ist abhängig vom *OnExecute*-Ereignis von *Form1.IdTCPServer1*:

```
if TabItem3.IsSelected then
Form3.Timer2.Enabled:=True;
```

Der Ende von *Timer2* ist abhängig vom *OnTimer*-Ereignis von *Timer2*:

```
procedure TForm3.Timer2Timer(Sender: TObject);
begin
Button5Click(self);
Timer2.Enabled:=False;
end;
```

Auf diese Weise kann *Button5Click*-Methoden einmal aufgerufen werden, wenn der Server jedes Mal antwortet, dann wird der nächste Befehl gesendet usw.

4.5.2.4 Multifunktionsrealisierung

Die Schließung des Fensters:

Nach dem Drücken der Schließen-Button wird das gesamte Fenster geschlossen. Das *OnClick*-Ereignis von Schließen-Button wird verwendet.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
```

Der Beginn der Aufgabe:

Nach dem Drücken der Starten-Button wird der Roboter in der Startposition bewegt und der Neigungsverlauf gestartet. Das *OnClick*-Ereignis von *Button2* wird verwendet.

```
procedure TForm3.Button2Click(Sender: TObject);
begin
```

```
Form1.hContext.Connection.IOHandler.WriteLn
('movel(p[300,0,400,-180,0,180],v=30)');
Timer1.Enabled:=True;
end;
```

Beenden des Roboterprogramms:

Nach dem Drücken der Beenden-Button wird der Roboterverlauf beendet. Das *OnClick*-Ereignis von Beenden-Button wird verwendet. Der Exit-Befehl wird gesendet.

```
procedure TForm3.Button3Click(Sender: TObject);
begin
Form1.hContext.Connection.IOHandler.WriteLn('exit');
end;
```

Die Löschung des Memoinhalts:

Nach dem Drücken der Clear-Button wird der Memoinhalt alles gelöscht. Das *OnClick*-Ereignis von Clear-Button wird verwendet.

```
procedure TForm3.Button4Click(Sender: TObject);
begin
Memo1.Lines.Clear;
end;
```

Beenden des Teilprogramms:

Nach dem Drücken der Stoppen-Button wird der Neigungsverlauf gestoppt. Das *OnClick*-Ereignis von Stoppen-Button wird verwendet.

```
procedure TForm3.Button6Click(Sender: TObject);
begin
Timer1.Enabled:=False;
Label2.Text := FloatToStr(0);
Label3.Text := FloatToStr(0);
end;
```

Neustarten des Teilprogramms:

Nach dem Drücken der Weiter-Button wird der Neigungsverlauf weiter ausführen. Das *OnClick*-Ereignis von Weiter-Button wird verwendet.

```
procedure TForm3.Button7Click(Sender: TObject);
begin
Timer1.Enabled:=True;
end;
```

Die Bewegung zum Startpunkts:

Nach dem Drücken der Startposition-Button wird der Roboter in der Startposition $[300, 0, 400, -180, 0, 180]$ bewegt. Das OnClick-Ereignis von Startposition-Button wird verwendet.

```
procedure TForm3.Button9Click(Sender: TObject);
begin
Form1.hContext.Connection.IOHandler.WriteLn
('movel(p[300,0,400,-180,0,180],v=30)');
end;
```

4.5.3 Systemtest

Die Bedienschnittstelle ist klar, leicht zu verstehen und zu bedienen und kann Aufgaben direkt ausführen.

① Herstellen eine Verbindung mit TCP/IP-Protokoll (Siehe Abb. 4.3-3)

② Befehltest:

a. Kippen das Tablet um einen beliebigen Winkel größer als $27°$ in X+Richtung und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

b. Kippen das Tablet um einen beliebigen Winkel größer als $27°$ in Y-Richtung und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

Der endgültige Operationseffekt ist wie folgt:

Abbildung 4.5-3: Das Befehltest_PositionX der Teilaufgabe 3(Tablet)

Abbildung 4.5-4: Das Befehltest_PositionY der Teilaufgabe 3(Tablet)

③Grenzwerttest:

a. Kippen das Tablet um einen beliebigen Winkel größer als $27°$ in X-Richtung bis $100\,mm$ und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

b. Kippen das Tablet um einen beliebigen Winkel größer als $27°$ in Y+Richtung bis $250\ mm$ und empfangen die Roboterantwort nach der Sendung der Steueranweisungen

Der endgültige Operationseffekt ist wie folgt:

Abbildung 4.5-5: Das Grenzwerttest_PositionX der Teilaufgabe 3(Tablet)

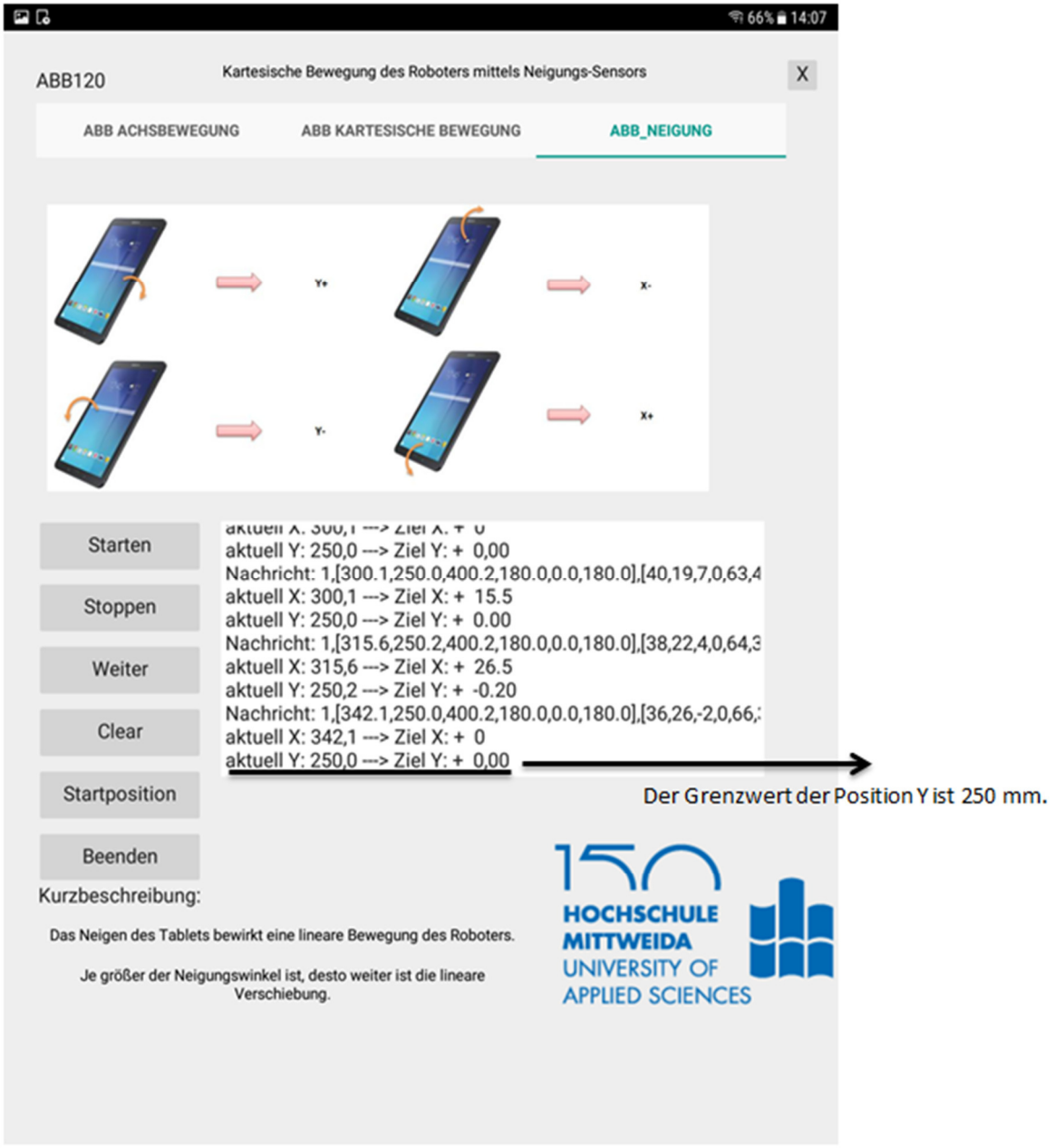

Abbildung 4.5-6: Das Grenzwerttest_PositionY der Teilaufgabe 3(Tablet)

④ Drücken den Beenden-Button und abbrechen die Verbindung (Siehe Abb. 4.3-3)

Testauswertung:

Die Kommunikation wird erfolgreich aufgebaut und abgebaut. Der Roboter bewegt sich in Echtzeit entsprechend der Neigung um die jeweilige Achse. Der Grenzwert in X- und Y-Richtung entsprechen dem Bereich. Nach dem Drücken der Startposition-Button wird der Roboter in den Startpunkt $[300, 0, 400, -180, 0, 180]$ bewegt.

4.6 Formenverbindung und Wechseln

Die drei Formen müssen miteinander in Zusammenhang und Verbindung gebracht werden. Dies wird realisiert durch TabControl und drei TabItems.

Abbildung 4.6-1: Die Bedienschnittstellen der Formenverbindung und Wechseln

Wenn Form1 angezeigt wird, wird *OnClick*-Ereignis von entsprechendem TabItem verwendet, um das Wechseln der Form zu steuern.

```
procedure TForm1.TabItem1Click(Sender: TObject);
begin
Form1.Show;
Form2.TabItem2.IsSelected:=True;
end;

procedure TForm1.TabItem2Click(Sender: TObject);
begin
Form2.Show;
Form2.TabItem2.IsSelected:=True;
end;

procedure TForm1.TabItem3Click(Sender: TObject);
begin
Form3.Show;
Form3.TabItem3.IsSelected:=True;
end;
```

Das Programmierungsabschnitt liegt in Form1. So gleich sind Form2 und Form3.

5 Zusammenfassung

Um die Flexibilität des Steuerungsendes und Kontrollerfahrung der traditionellen PC-Fernbedienung des Roboters zu verbessern, wurde ein Roboter-Fernsteuerungssystem basierend auf Android-Mobile-Intelligent-Terminal entwickelt.

Zuerst wurde die Systemstruktur analysiert und der Entwicklungsplan erstellt. Dann Android-Client, Roboter-Server und Robotersteuerprogramm entwickelt. Danach wurde das Designkonzept eines Roboter-Fernsteuerungssystems entwickelt, das auf der Android-Plattform basiert, d.h. verwendet ein Mobiltelefon oder einen Tablet-Computer mit einer Android-Plattform über ein drahtloses Netzwerk, um die Fernbedienung von Robotern zu erreichen. Dann wurde die Struktur und das Steuerungsprinzip des Fernsteuerungssystems analysiert.

Das Fernsteuerungssystem verwendete den C / S-Modus, d.h. den Client / Server-Modus. Das System bestand aus einem Mobile-Smart-Terminal-basierten Client und einem Roboter-basierten Server. Der Client verband sich über ein drahtloses lokales Netzwerk (WLAN) mit dem Server. Das System umfasste roboterbasierter Server und Android-basierte Client. Der Server des Systems war in ABBScript-Sprache geschrieben, der Client wurde durch Delphi verwirklicht. Am Ende konnte das Android-Tablet zur Fernsteuerung des Roboters verwendet werden.

Nach dem Test erfüllte das Steuerungssystem die Erwartungen. Der Roboter konnte sich korrekt mit der Steuerungssoftware verbinden. Nach der Anweisung wurde die entsprechende Bewegung in Echtzeit in einem bestimmten Bereich ausgeführt und die entsprechende Antwort wurde empfangen und ausgewertet.

6 Ausblick

Das Fernsteuerungssystem des Roboters basiert auf der Android-Plattform, die in dieser Arbeit entwickelt wurde, nur auf dem Prototyp des beweglichen Roboterentwicklungssystems des intelligenten Terminalendroboters.

Human-Computer-Interaktionsschnittstelle hat auch viel Raum für Verbesserungen. Wenn sich die Schnittstelle ändert und mehrere Funktionen hinzufügen, wird auch der Steuerungsaspekt aktualisiert. Die Realisierung der Fernsteuerung in einem WLAN ist nur der erste Schritt und wird in Zukunft auf das WAN (Wide Area Network) ausgedehnt. Bis dahin werden Fernsteuerungssysteme größere Herausforderungen in Echtzeit haben. Als Reaktion auf die oben genannten Mängel und Erweiterungen wird das System in zukünftigen Arbeiten optimiert und verbessert.

Die erfolgreiche Entwicklung des auf der Android-Plattform basierenden Roboterfernsteuerungssystems zeigt, dass das mobile intelligente Terminal in Kombination mit dem Fernsteuerungsmodus des Roboters möglich ist.

Bei der kontinuierlichen Verbesserung der Hardwareleistung wird davon ausgegangen, dass sich mobile intelligente Endgeräte in den nächsten Jahren mit einer alarmierenden Geschwindigkeit entwickeln werden. Dies betrifft nicht nur die herkömmlichen Endgeräte, sondern auch die Auswirkungen auf den herkömmlichen Fernsteuerungsmodus. Es ist sicher, dass mobile intelligente Terminals in näher Zukunft stärker in der wissenschaftlichen Forschung eingesetzt werden. Es wird auch immer mehr mobile Fernsteuerungssysteme geben, die auf mobilen intelligenten Terminals basieren, die in Produktion und Leben entwickelt und eingesetzt werden.

Der Fernsteuerroboter für intelligente Terminals wird zu einem Hotspot im Zukunftsforschungsbereich.

Literatur

[1] Bloch, M.; Reyhanoglu, M.; McClamroch, H.: Control and stabilization of nonholonomic dynamic systems. IEEE Transactions on Automatic Control, 37(11):1746-1757, 1992.

[2] Baillieul, J.: Geometric methods for nonlinear optimal control problems. Journal of Optimization Theory and Applications, 25(4):519–548, 1978.

[3] Astrom, K-J. ; Wittenmark, B.: Computer Controlled Systems: Theory and Design. Prentice-Hall, 1984.

[4] Brockett, W-R. ; Stokes, A. ; Park, F.: A geometrical formulation of the dynamical equations describing kinematic chains. In IEEE International Conference on Robotics and Automation, pages 637–642, 1993.

[5] Fernandes, C. ; Gurvits, L. ; Attitude, Z.:control of a space platform manipulator system using internal motion. International Journal of Robotics Research, 1994.

[6] Stewart, D.: A platform with six degrees of freedom. Proceedings of the Institute of Mechanical Engineering, 180, part I(5):371-186, 1954. 1965-66.

[7] Ferworn, R. ;Vecchia, I.:Wireless teleoperation via the World Wide Web Proc.1ASTED Conf. On Robotice&Applications, santa Barbara, 1999.

[8] Nehmazow, U. ; Buhlmeier, A. ; Durer, H. ; Nolte, M.: Remote control of mobile robot Via Internet, Dept.of Computer Science, University of Manchester, Technical ReportSeries, UMCS-96-2-3.

[9] Ferworn, R. ; Vecchia, I.: Wireless teleoperation via the World Wide Web, Proc.1ASTED Conf.On Robotice&Applications, santa Barbara, 1999.

[10] Simmons, R.: An autonomous mobile robots on the web, Proceedings ofIROS'98Worldshop onWeb Robots, page43-48, Victoria, Canada 12-17 October 1998.

[11] Saucy, P. ; Mondana, F.: Open access tO a mobile robot on the Intemet, IEEE Robotics and Automation Magazine, 2000.

[12] Schulz, D. ; Burgard, W. ; Fox, D.: Web interface for mobile robots in public places, IEEE Robotics and Automation Magazine, page 48-56, March 2000.

[13] Bauer, E. ; Schitiu, M. ; Lorand, C. ; Premaratne, K.: Total Delay Compensation in LAN Control Systems and Implications for Scheduling. Proceeding of the American Control Conference 2001,6:4300-4305.

[14] Produktspezifikation IRB 120, verfügbar am 06.06.2018

[15] Fiorini, P. ; Oboe, R.: Internet-based telerobotics and approaches.International Conference of Advanced Robotics, 1997, 7: 765-770.

Anlagen

Anlagen, ABBScript-Sprache

Ausgang setzen:

-set_digital_output(<Nummer>,<Wert>)
-Nummer: Ganzzahl (1-3)
-Wert: „True“ oder „true“ für 1
„False“ oder „false“ für 0
-Beispiel:
set_digital_output(2,True)

Achsbewegung:

-movej([<Achse1>,<Achse2>,<Achse3>,<Achse4>,<Achse5>,<Achse6>],v=<Gs.>)
-Achse: Achswinkel im Bogenmaß [**rad**]
-Gs.: Geschwindigkeit [**mm/s**], bei keiner Angabe werden 50 mm/s verwendet
-maximal **6** Nachkommastellen verwenden!
-Beispiel:
movej([0.123,-0.2651,0.3148,-3.0386,-1.3935,0],v=15)

Linearbewegung:

-zu einem festen Punkt
-movel(p[<X>,<Y>,<Z>,<A>,<B>,<C>],v=<Gs.>)
-X,Y,Z: Kartesische Koordinaten in Millimeter [**mm**]
-A,B,C: Eulerwinkel im Grad [**°**]
-Gs.: Geschwindigkeit [**mm/s**], bei keiner Angabe werden 50 mm/s verwendet
-Als Koordinatensystem wird immer wobj0 (Weltkoordinatensystem) genutzt
-Als Werkzeug wird *tool0* (Flanschkoordinaten) genutzt
-maximal **3** Nachkommastellen verwenden!
-Beispiel:
movel(p[100.1,-200.3,300.4,-180,0,180],v=25)

-zu einem Offset eines festen Punktes
-movel(pose_trans([<X1>,<Y1>,<Z1>,<A1>,<B1>,<C1>],[<X2>,<Y2>,<Z2>,0,0,0]),v=<Gs.>
-X1,Y1,Z1: Kartesische Koordinaten des Ursprungspunktes in Millimeter [**mm**]
-A1,B1,C1: Eulerwinkel des Ursprungpunktes im Grad [**°**]
-X2,Y2,Z2: Kartesische Verschiebung in Millimeter [**mm**]
-Gs.: Geschwindigkeit [**mm/s**], bei keiner Angabe werden 50 mm/s verwendet
-Als Koordinatensystem wird immer *wobj0* (Weltkoordinatensystem) genutzt
-Als Werkzeug wird *tool0* (Flanschkoordinaten) genutzt
-Keine Nachkommastellen verwenden!
-Beispiel:
movel(pose_trans([123,-251,48,-180,0,180],[50,-100,20,0,0,0]),v=32)

-zu einem Offset der aktuellen Position
-movel(pose_trans(get_actual_tcp_pose(),[<X>,<Y>,<Z>,0,0,0]),v=<Gs.>)
- X,Y,Z: Kartesische Verschiebung in Millimeter [**mm**]
-Gs.: Geschwindigkeit [**mm/s**], bei keiner Angabe werden 50 mm/s verwendet
-Als Koordinatensystem wird immer *wobj0* (Weltkoordinatensystem) genutzt
-Als Werkzeug wird *tool0* (Flanschkoordinaten) genutzt
-maximal **2** Nachkommastellen verwenden!
-Beispiel:
movel(pose_trans(get_actual_tcp_pose(),[0.1,0.2,-0.3,0,0,0]),v=50)

Pfadaufruf:

-Aufruf eines einprogrammierten Pfades
-„path_<Nummer>“
-Nummer: Nummer des aufzurufenden Pfades
-Beispiel:
path_02

Beenden

-„exit“
-das Roboterprogramm wird beendet
-Beispiel:
Exit

Systemantwort

-Die Steuerung wird auf jeden Befehl mit einem String antworten
-Beispiel:
01,[100.1,-200.1,300.1,-180,0,-180],[-180,-180,-180,-180,-180,-180]
-Inhalt:
<Status>,[<X>,<Y>,<Z>,<A>,<B>,<C>],[<Ax1>,<Ax2>,<Ax3>,<Ax4>,<Ax5>,<Ax6>]
-Status:
00:
01:
02:
-X,Y,Z: Kartesische Koordinaten (TCP) in Millimeter [**mm**] (max. 1 Nachkomma)
-A,B,C: Eulerwinkel (TCP) im Grad [°] (x,y,z) (max. 1 Nachkomma)
-Ax*n*: Achswinkel *n* in Grad [°] (keine Nachkomma)
-Als Koordinatensystem wird immer *wobj0* (Weltkoordinatensystem) genutzt
-Als Werkzeug wird *tool0* (Flanschkoordinaten) genutzt

Anlagen, String-Antwort zerschneiden

Ein Beispiel für die Roboterantworten:

$1, [302.0,0.0,558.0, -180.0,0.0,180.0], [winkel1, winkel2, winkel3, winkel4, winkel5, winkel6]$

Die Schritt-für-Schritt Analyse ist wie folgt:

```
z:=IntToStr(pos(']',Antwort));
      // die Position der ersten rechts eckigen Klammer in der Antwort
a:=copy(Antwort,StrToInt(z)+2,length(Antwort)-StrToInt(z)-1);
      // a = [winkel1, winkel2, winkel3, winkel4, winkel5, winkel6]
b:=IntToStr(pos(',',a));
      // die Position des ersten Kommas in String a
eins:=copy(a,2,StrToInt(b)-2);
      // eins = winkel1
c:=copy(a,StrToInt(b)+1,length(a)-StrToInt(b));
      // c = winkel2, winkel3, winkel4, winkel5, winkel6]
d:=IntToStr(pos(',',c));
      // die Position des ersten Kommas in String c
zwei:=copy(c,1,StrToInt(d)-1);
      // zwei = winkel2
e:=copy(c,StrToInt(d)+1,length(c)-StrToInt(d));
      // e = winkel3, winkel4, winkel5, winkel6]
f:=IntToStr(pos(',',e));
      // die Position des ersten Kommas in String e
drei:=copy(e,1,StrToInt(f)-1);
      // drei = winkel3
g:=copy(e,StrToInt(f)+1,length(e)-StrToInt(f));
      // g = winkel4, winkel5, winkel6]
h:=IntToStr(pos(',',g));
      // die Position des ersten Kommas in String g
vier:=copy(g,1,StrToInt(h)-1);
      // vier = winkel4
i:=copy(g,StrToInt(h)+1,length(g)-StrToInt(h));
      // i = winkel5, winkel6]
j:=IntToStr(pos(',',i));
      // die Position des ersten Kommas in String i
funf:=copy(i,1,StrToInt(j)-1);
      // funf = winkel5
k:=copy(i,StrToInt(j)+1,length(i)-StrToInt(j));
      // k = winkel6]
sechs:=copy(k,1,length(k)-1);
      // sechs = winkel6
```

Anlagen, Programmierung_Form1

Unit_ABBachsbewegung in RAD Studio 10.2:

```
unit ABBachsbewegung;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, Sys-
  tem.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,
  FMX.Dialogs, FMX.StdCtrls, FMX.Controls.Presentation, FMX.ScrollBox,
  FMX.Memo, IdBaseComponent, IdComponent, IdCustomTCPServer, IdTCPServer,
  IdContext, FMX.Layouts, FMX.Edit, FMX.Objects, FMX.TabControl;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    ScrollBar1: TScrollBar;
    ScrollBar2: TScrollBar;
    ScrollBar3: TScrollBar;
    ScrollBar4: TScrollBar;
    ScrollBar5: TScrollBar;
    ScrollBar6: TScrollBar;
    Memo1: TMemo;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Rectangle1: TRectangle;
    Rectangle2: TRectangle;
    Rectangle3: TRectangle;
    Rectangle4: TRectangle;
    Rectangle5: TRectangle;
    Rectangle6: TRectangle;
    IdTCPServer1: TIdTCPServer;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
```

```
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    RadioButton5: TRadioButton;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    TabControl1: TTabControl;
    TabItem1: TTabItem;
    TabItem2: TTabItem;
    TabItem3: TTabItem;
    Image1: TImage;
    Text1: TText;
    Label14: TLabel;
    Text2: TText;
    Text3: TText;
    Button5: TButton;
    procedure IdTCPServer1Connect(AContext: TIdContext);
    procedure IdTCPServer1Disconnect(AContext: TIdContext);
    procedure IdTCPServer1Exception(AContext: TIdContext;
      AException: Exception);
    procedure IdTCPServer1Execute(AContext: TIdContext);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
    procedure ScrollBar2Change(Sender: TObject);
    procedure ScrollBar3Change(Sender: TObject);
    procedure ScrollBar4Change(Sender: TObject);
    procedure ScrollBar5Change(Sender: TObject);
    procedure ScrollBar6Change(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure Edit5Change(Sender: TObject);
    procedure Edit6Change(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure RadioButton4Click(Sender: TObject);
    procedure RadioButton5Click(Sender: TObject);
    procedure CheckBox1Change(Sender: TObject);
    procedure CheckBox2Change(Sender: TObject);
    procedure CheckBox3Change(Sender: TObject);
    procedure TabItem1Click(Sender: TObject);
    procedure TabItem2Click(Sender: TObject);
```

```
    procedure TabItem3Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private-Deklarationen }
    z, a, b, c, d, e, f, g, h, i, j, k, eins, zwei, drei, vier, funf,
sechs: String;
    txt1, txt2, txt3, txt4, txt5, txt6: string;
  public
    { Public-Deklarationen }
    hContext: TIdContext;
    Antwort: string;
  end;

var
  Form1: TForm1;
  gesch: String = '10';

implementation

{$R *.fmx}

uses ABBKartesisch, ABB_Neigung;

procedure TForm1.Button1Click(Sender: TObject);
begin
   Form1.Close;
end;
      // Schließen-Button einstellen

procedure TForm1.Button2Click(Sender: TObject);
begin
  txt1 := FloatToStrF(scrollbar1.Value*3.1415/180, ffFixed, 8, 6);
  txt2 := FloatToStrF(scrollbar2.Value*3.1415/180, ffFixed, 8, 6);
  txt3 := FloatToStrF(scrollbar3.Value*3.1415/180, ffFixed, 8, 6);
  txt4 := FloatToStrF(scrollbar4.Value*3.1415/180, ffFixed, 8, 6);
  txt5 := FloatToStrF(scrollbar5.Value*3.1415/180, ffFixed, 8, 6);
  txt6 := FloatToStrF(scrollbar6.Value*3.1415/180, ffFixed, 8, 6);
      // Scrollbar.Value in Bogenmaß in String anzeigen und 6 Nachkommas-
      tellen verwenden
  txt1:=StringReplace (txt1, ',', '.', []) ;
  txt2:=StringReplace (txt2, ',', '.', []) ;
  txt3:=StringReplace (txt3, ',', '.', []) ;
  txt4:=StringReplace (txt4, ',', '.', []) ;
  txt5:=StringReplace (txt5, ',', '.', []) ;
  txt6:=StringReplace (txt6, ',', '.', []) ;
      // StringReplace im txt1-6 das Komma durch den Punkt ersetzen, da-
      mit die Robotersteuerung der Befehl erkennt
  hContext.Connection.IOHandler.WriteLn('movej(['+txt1+', '+txt2+',
'+txt3+', '+txt4+', '+txt5+', '+txt6+'],v='+gesch+')');
end;
      // Senden-Button einstellen

procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
   hContext.Connection.IOHandler.WriteLn('exit');
end;
      // Beenden-Button einstellen

procedure TForm1.Button4Click(Sender: TObject);
begin
   Memo1.Lines.Clear;
end;
      // Clear-Button einstellen

procedure TForm1.Button5Click(Sender: TObject);
begin
   hContext.Connection.IOHandler.WriteLn('movej([0,0,0,0,1.57,0],v=50)');
end;
      // Grundposition-Button einstellen

procedure TForm1.CheckBox1Change(Sender: TObject);
begin
   if CheckBox1.IsChecked=True then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(1,True)');
   if CheckBox1.IsChecked=False then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(1,False)');
end;
      // rote Lampe an und aus einstellen

procedure TForm1.CheckBox2Change(Sender: TObject);
begin
   if CheckBox2.IsChecked=True then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(2,True)');
   if CheckBox2.IsChecked=False then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(2,False)');
end;
      // gelbe Lampe an und aus einstellen

procedure TForm1.CheckBox3Change(Sender: TObject);
begin
   if CheckBox3.IsChecked=True then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(3,True)');
   if CheckBox3.IsChecked=False then
   hContext.Connection.IOHandler.WriteLn('set_digital_output(3,False)');
end;
      // grüne Lampe an und aus einstellen

procedure TForm1.Edit1Change(Sender: TObject);
begin
   Scrollbar1.Value :=StrToFloat(Edit1.Text);
end;
      // Edit1.Text in Scrollbar1.Value anzeigen

procedure TForm1.Edit2Change(Sender: TObject);
begin
```

```
   Scrollbar2.Value :=StrToFloat(Edit2.Text);
end;
      // Edit2.Text in Scrollbar2.Value anzeigen

procedure TForm1.Edit3Change(Sender: TObject);
begin
   Scrollbar3.Value :=StrToFloat(Edit3.Text);
end;
      // Edit3.Text in Scrollbar3.Value anzeigen

procedure TForm1.Edit4Change(Sender: TObject);
begin
   Scrollbar4.Value :=StrToFloat(Edit4.Text);
end;
      // Edit4.Text in Scrollbar4.Value anzeigen

procedure TForm1.Edit5Change(Sender: TObject);
begin
   Scrollbar5.Value :=StrToFloat(Edit5.Text);
end;
      // Edit5.Text in Scrollbar5.Value anzeigen

procedure TForm1.Edit6Change(Sender: TObject);
begin
   Scrollbar6.Value :=StrToFloat(Edit6.Text);
end;
      // Edit6.Text in Scrollbar6.Value anzeigen

procedure TForm1.IdTCPServer1Connect(AContext: TIdContext);
begin
   Memo1.Lines.Add('Verbindung: ' + AContext.Binding.PeerIP);
   hContext := AContext;
   CheckBox1.IsChecked:=True;
   CheckBox3.IsChecked:=True;
   TabItem1.IsSelected:=True;
end;
      // Anfangsstatus:
      rote Lampe an, grüne Lampe an, TabItem1 Ausgewählt
      Beim OnConnect-Ereignis nach der Verbindung wird die IP-Adresse in
      Memo1 angezeigt.

procedure TForm1.IdTCPServer1Disconnect(AContext: TIdContext);
begin
   Memo1.Lines.Add('Verbindungsabbruch: ' + AContext.Binding.PeerIP);
end;
      // Beim OnDisconnect-Ereignis nach dem Verbindungsabbruch wird auch
      die IP-Adresse in Memo1 zeigt.

procedure TForm1.IdTCPServer1Exception(AContext: TIdContext;
  AException: Exception);
begin
   showmessage('Fehler')
```

```
end;
      // Beim OnException-Ereignis zeigt in der Bedienschnittstelle das
      Wort Fehler, wenn die Lese- oder Schreibdaten nicht erfolgreich
      ausgeführt werden.

procedure TForm1.IdTCPServer1Execute(AContext: TIdContext);
begin
   Antwort:=AContext.Connection.IOHandler.ReadLn;
   Memo1.Lines.Add('Nachricht: ' + Antwort);

   z:=IntToStr(pos(']',Antwort));
   a:=copy(Antwort,StrToInt(z)+2,length(Antwort)-StrToInt(z)-1);
   b:=IntToStr(pos(',',a));
   eins:=copy(a,2,StrToInt(b)-2);
   c:=copy(a,StrToInt(b)+1,length(a)-StrToInt(b));
   d:=IntToStr(pos(',',c));
   zwei:=copy(c,1,StrToInt(d)-1);
   e:=copy(c,StrToInt(d)+1,length(c)-StrToInt(d));
   f:=IntToStr(pos(',',e));
   drei:=copy(e,1,StrToInt(f)-1);
   g:=copy(e,StrToInt(f)+1,length(e)-StrToInt(f));
   h:=IntToStr(pos(',',g));
   vier:=copy(g,1,StrToInt(h)-1);
   i:=copy(g,StrToInt(h)+1,length(g)-StrToInt(h));
   j:=IntToStr(pos(',',i));
   funf:=copy(i,1,StrToInt(j)-1);
   k:=copy(i,StrToInt(j)+1,length(i)-StrToInt(j));
   sechs:=copy(k,1,length(k)-1);
      // String-Antwort zerschneiden
      // Kommas suchen
      // Siehe Anlage, String-Antowort zerschneiden

   Memo1.Lines.Add('Achswinkel: ' + a);
   Scrollbar1.Value :=StrToFloat(eins);
   Scrollbar2.Value :=StrToFloat(zwei);
   Scrollbar3.Value :=StrToFloat(drei);
   Scrollbar4.Value :=StrToFloat(vier);
   Scrollbar5.Value :=StrToFloat(funf);
   Scrollbar6.Value :=StrToFloat(sechs);

      // Achswerte mit StringToInteger oder StringToFloat umwandeln
      // ScrollBar.Value:= Achswert

   Form2.Button8Click(self);
   if TabItem3.IsSelected then
   Form3.Timer2.Enabled:=True;
end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
   gesch:='10';
end;
```

```
      // die Geschwindigkeitsklasse 10 mm/s einstellen

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
   gesch:='30';
end;
      // die Geschwindigkeitsklasse 30 mm/s einstellen

procedure TForm1.RadioButton3Click(Sender: TObject);
begin
   gesch:='50';
end;
      // die Geschwindigkeitsklasse 50 mm/s einstellen

procedure TForm1.RadioButton4Click(Sender: TObject);
begin
   gesch:='70';
end;
      // die Geschwindigkeitsklasse 70 mm/s einstellen

procedure TForm1.RadioButton5Click(Sender: TObject);
begin
   gesch:='100';
end;
      // die Geschwindigkeitsklasse 100 mm/s einstellen

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
   Label8.Text := FloatToStrF(scrollbar1.Value, ffFixed, 8, 6);
end;
      // Scrollbar1.Value in Grad in String in Label8.Text anzeigen und 6
      Nachkommastellen verwenden

procedure TForm1.ScrollBar2Change(Sender: TObject);
begin
   Label9.Text := FloatToStrF(scrollbar2.Value, ffFixed, 8, 6);
end;
      // Scrollbar2.Value in Grad in String in Label9.Text anzeigen und 6
      Nachkommastellen verwenden

procedure TForm1.ScrollBar3Change(Sender: TObject);
begin
   Label10.Text := FloatToStrF(scrollbar3.Value, ffFixed, 8, 6);
end;
      // Scrollbar3.Value in Grad in String in Label10.Text anzeigen und
      6 Nachkommastellen verwenden

procedure TForm1.ScrollBar4Change(Sender: TObject);
begin
   Label11.Text := FloatToStrF(scrollbar4.Value, ffFixed, 8, 6);
end;
      // Scrollbar4.Value in Grad in String in Label11.Text anzeigen und
      6 Nachkommastellen verwenden
```

```
procedure TForm1.ScrollBar5Change(Sender: TObject);
begin
   Label12.Text := FloatToStrF(scrollbar5.Value, ffFixed, 8, 6);
end;
      // Scrollbar5.Value in Grad in String in Label12.Text anzeigen und
      6 Nachkommastellen verwenden

procedure TForm1.ScrollBar6Change(Sender: TObject);
begin
   Label13.Text := FloatToStrF(scrollbar6.Value, ffFixed, 8, 6);
end;
      // Scrollbar6.Value in Grad in String in Label13.Text anzeigen und
      6 Nachkommastellen verwenden

procedure TForm1.TabItem1Click(Sender: TObject);
begin
   Form1.Show;
   Form1.TabItem1.IsSelected:=True;
end;
      // selbst anzeigen

procedure TForm1.TabItem2Click(Sender: TObject);
begin
   Form2.Show;
   Form2.TabItem2.IsSelected:=True;
end;
      // das Wechseln zwischen Form1 und Form2 realisieren

procedure TForm1.TabItem3Click(Sender: TObject);
begin
   Form3.Show;
   Form3.TabItem3.IsSelected:=True;
end;
      // das Wechseln zwischen Form1 und Form3 realisieren

end.
```

Anlagen, Programmierung_Form2

Unit_ ABBKartesisch in RAD Studio 10.2:

```
unit ABBKartesisch;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, Sys-
  tem.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,
  FMX.Dialogs, FMX.TabControl, FMX.Controls.Presentation, FMX.StdCtrls,
  FMX.Objects, IdBaseComponent, IdComponent, IdCustomTCPServer, IdTCPSer-
  ver, IdContext, FMX.ScrollBox, FMX.Memo, FMX.Edit;

type
  TForm2 = class(TForm)
    TabControl1: TTabControl;
    TabItem1: TTabItem;
    TabItem2: TTabItem;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    Image5: TImage;
    Image6: TImage;
    RadioButton1: TRadioButton;
    Memo1: TMemo;
    Button8: TButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    RadioButton5: TRadioButton;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    RadioButton9: TRadioButton;
    RadioButton10: TRadioButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
```

```
    Label6: TLabel;
    Label7: TLabel;
    Button9: TButton;
    Button10: TButton;
    TabItem3: TTabItem;
    Image7: TImage;
    Label8: TLabel;
    Text1: TText;
    Text2: TText;
    Text3: TText;
    Button11: TButton;
    procedure TabItem1Click(Sender: TObject);
    procedure TabItem2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure RadioButton4Click(Sender: TObject);
    procedure RadioButton5Click(Sender: TObject);
    procedure RadioButton6Click(Sender: TObject);
    procedure RadioButton7Click(Sender: TObject);
    procedure RadioButton8Click(Sender: TObject);
    procedure RadioButton9Click(Sender: TObject);
    procedure RadioButton10Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure TabItem3Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
  private
    { Private-Deklarationen }
    a, b, c, d, e, f, g, h: String;
  public
    { Public-Deklarationen }
    Antwort: String;
  end;

var
  Form2: TForm2;
  S: String = '10';
  gesch2: String = '10';
  eins: String = '302';
  zwei: String = '0';
  drei: String = '558';

implementation
```

```
{$R *.fmx}

uses ABBachsbewegung, ABB_Neigung;

procedure TForm2.Button10Click(Sender: TObject);
begin
   Memo1.Lines.Clear;
end;
      // Clear-Button einstellen

procedure TForm2.Button11Click(Sender: TObject);
begin
   Form1.hContext.Connection.IOHandler.WriteLn
      ('movej([0,0,0,0,1.57,0],v=50)');
end;
      // Grundposition-Button einstellen

procedure TForm2.Button1Click(Sender: TObject);
begin
   Form1.Close;
end;
      // Schließen-Button einstellen

procedure TForm2.Button2Click(Sender: TObject);
begin
   zwei:=StringReplace (zwei, '.', ',', []) ;
      // StringReplace in der Y-Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen
   if (StrToFloat(zwei)>-250) and (StrToFloat(zwei)-StrToFloat(S)>=-250)
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),[0,-'+S+',0,0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_Y- auch mehr
      als -250 mm, dann bewegt der Roboter in X- in -S mm

   else if (StrToFloat(zwei)>-250)
   and (StrToFloat(zwei)-StrToFloat(S)<-250) then
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),[0,'+FloatToStr(-250-
      StrToFloat(zwei))+',0,0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_Y- weniger als
      -250 mm, dann bewegt der Roboter in Y- direkt bis Minimalwert

end;

procedure TForm2.Button3Click(Sender: TObject);
begin
   eins:=StringReplace (eins, '.', ',', []) ;
      // StringReplace in der X-Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen

   if (StrToFloat(eins)>100) and (StrToFloat(eins)-StrToFloat(S)>=100)
```

```
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),[-'+S+',0,0,0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_X- auch mehr
      als 100 mm, dann bewegt der Roboter in X- in -S mm

   else if (StrToFloat(eins)>100)
   and (StrToFloat(eins)-StrToFloat(S)<100) then
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),['+FloatToStr(100-
      StrToFloat(eins))+',0,0,0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_X- weniger als
      100 mm, dann bewegt der Roboter in X- direkt bis Minimalwert

end;

procedure TForm2.Button4Click(Sender: TObject);
begin
   eins:=StringReplace (eins, '.', ',', []) ;
      // StringReplace in der X+Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen

   if (StrToFloat(eins)<400) and (StrToFloat(eins)+StrToFloat(S)<=400)
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),['+S+',0,0,0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_X+ auch inner-
      halb 400 mm, dann bewegt der Roboter in X+ in S mm

   else if (StrToFloat(eins)<400)
   and (StrToFloat(eins)+StrToFloat(S)>400) then
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),['+FloatToStr(400-
      StrToFloat(eins))+',0,0,0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_X+ mehr als 400
      mm, dann bewegt der Roboter in X+ direkt bis Maximalwert

end;

procedure TForm2.Button5Click(Sender: TObject);
begin
   zwei:=StringReplace (zwei, '.', ',', []) ;
      // StringReplace in der Y+Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen

   if (StrToFloat(zwei)<250) and (StrToFloat(zwei)+StrToFloat(S)<=250)
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),[0,'+S+',0,0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_Y+ auch inner-
      halb 250 mm, dann bewegt der Roboter in Y+ in S mm

   else if (StrToFloat(zwei)<250)
   and (StrToFloat(zwei)+StrToFloat(S)>250) then
```

```
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),[0,'+FloatToStr(250-
      StrToFloat(zwei))+',0,0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_Y+ mehr als 250
      mm, dann bewegt der Roboter in Y+ direkt bis Maximalwert

end;

procedure TForm2.Button6Click(Sender: TObject);
begin
   drei:=StringReplace (drei, '.', ',', []) ;
      // StringReplace in der Z+Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen

   if (StrToFloat(drei)<600) and (StrToFloat(drei)+StrToFloat(S)<=600)
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),[0,0,'+S+',0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_Z+ auch inner-
      halb 600 mm, dann bewegt der Roboter in X+ in S mm

   else if (StrToFloat(drei)<600)
   and (StrToFloat(drei)+StrToFloat(S)>600) then
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),[0,0,'+FloatToStr(600-
      StrToFloat(drei))+',0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_Z+ mehr als 600
      mm, dann bewegt der Roboter in Z+ direkt bis Maximalwert

end;

procedure TForm2.Button7Click(Sender: TObject);
begin
   drei:=StringReplace (drei, '.', ',', []) ;
      // StringReplace in der Z-Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen

   if (StrToFloat(drei)>200) and (StrToFloat(drei)-StrToFloat(S)>=200)
   then
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),[0,0,-'+S+',0,0,0]),v='+gesch2+')')
      // Nach der Verschiebung wird die aktuelle Position_Z- auch mehr
      als 200 mm, dann bewegt der Roboter in Z- in -S mm

   else if (StrToFloat(drei)>200)
   and (StrToFloat(drei)-StrToFloat(S)<200) then
      Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(pose_trans(get_actual_tcp_pose(),[0,0,'+FloatToStr(200-
      StrToFloat(drei))+',0,0,0]),v='+gesch2+')');
      // Nach der Verschiebung wird die aktuelle Position_Z- weniger als
      200 mm, dann bewegt der Roboter in Z- direkt bis Minimalwert

end;
```

```
procedure TForm2.Button8Click(Sender: TObject);
begin
   a:=IntToStr(pos('[',Form1.Antwort));
   b:=IntToStr(pos(']',Form1.Antwort));
   c:=copy(Form1.Antwort,StrToInt(a),StrToInt(b)-2);
   d:=IntToStr(pos(',',c));
   eins:=copy(c,2,StrToInt(d)-2);
   e:=copy(c,StrToInt(d)+1,length(c)-StrToInt(d));
   f:=IntToStr(pos(',',e));
   zwei:=copy(e,1,StrToInt(f)-1);
   g:=copy(e,StrToInt(f)+1,length(e)-StrToInt(f));
   h:=IntToStr(pos(',',g));
   drei:=copy(g,1,StrToInt(h)-1);

   Memo1.Lines.Add('Nachricht: ' + Form1.Antwort);
   Memo1.Lines.Add('Position+Eulerwinkel: ' + c);

   Label5.Text := eins;   // X-Position anzeigen
   Label6.Text := zwei;   // Y-Position anzeigen
   Label7.Text := drei;   // Z-Position anzeigen
end;

procedure TForm2.Button9Click(Sender: TObject);
begin
   Form1.hContext.Connection.IOHandler.WriteLn('exit');
end;
      // Beenden-Button einstellen

procedure TForm2.RadioButton10Click(Sender: TObject);
begin
   gesch2:='100';
end;
      // die Geschwindigkeitsklasse 100 mm/s einstellen

procedure TForm2.RadioButton1Click(Sender: TObject);
begin
   S:='10';
end;
      // die Verschiebungsklasse 10 mm/s einstellen

procedure TForm2.RadioButton2Click(Sender: TObject);
begin
   S:='20';
end;

procedure TForm2.RadioButton3Click(Sender: TObject);
begin
   S:='30';
end;

procedure TForm2.RadioButton4Click(Sender: TObject);
begin
```

```
   S:='50';
end;

procedure TForm2.RadioButton5Click(Sender: TObject);
begin
   S:='70';
end;

procedure TForm2.RadioButton6Click(Sender: TObject);
begin
   gesch2:='10';
end;

procedure TForm2.RadioButton7Click(Sender: TObject);
begin
   gesch2:='30';
end;

procedure TForm2.RadioButton8Click(Sender: TObject);
begin
   gesch2:='50';
end;

procedure TForm2.RadioButton9Click(Sender: TObject);
begin
   gesch2:='70';
end;

procedure TForm2.TabItem1Click(Sender: TObject);
begin
   Form1.Show;
   Form1.TabItem1.IsSelected:=True;
end;
      // das Wechseln zwischen Form2 und Form1 realisieren

procedure TForm2.TabItem2Click(Sender: TObject);
begin
   Form2.Show;
   Form2.TabItem2.IsSelected:=True;
end;
      // selbst anzeigen

procedure TForm2.TabItem3Click(Sender: TObject);
begin
   Form3.Show;
   Form3.TabItem3.IsSelected:=True;
end;
      // das Wechseln zwischen Form2 und Form3 realisieren

end.
```

Anlagen, Programmierung_Form3

Unit_ABB_Neigung in RAD Studio 10.2:

```
unit ABB_Neigung;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, Sys-
  tem.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,
  FMX.Dialogs, FMX.StdCtrls, FMX.TabControl, FMX.Controls.Presentation,
  FMX.ScrollBox, FMX.Memo, System.Sensors, System.Sensors.Components,
  FMX.Objects;

type
  TForm3 = class(TForm)
    Label1: TLabel;
    TabControl1: TTabControl;
    TabItem1: TTabItem;
    TabItem2: TTabItem;
    TabItem3: TTabItem;
    Button1: TButton;
    Button2: TButton;
    Memo1: TMemo;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Timer1: TTimer;
    MotionSensor1: TMotionSensor;
    Label2: TLabel;
    Label3: TLabel;
    Button6: TButton;
    Timer2: TTimer;
    Button7: TButton;
    Button8: TButton;
    Image1: TImage;
    Label4: TLabel;
    Text1: TText;
    Text2: TText;
    Text3: TText;
    Image2: TImage;
    Image3: TImage;
    Button9: TButton;
    procedure Button1Click(Sender: TObject);
    procedure TabItem1Click(Sender: TObject);
    procedure TabItem2Click(Sender: TObject);
    procedure TabItem3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
```

```
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
  private
    { Private-Deklarationen }
    a,b,c,d,e,f,g,h:String;
    eins,zwei,drei,eins2,zwei2:String;
  public
    { Public-Deklarationen }
    Antwort: String;
  end;

var
  Form3: TForm3;
  Runde: Integer = 0;

implementation

{$R *.fmx}

uses ABBachsbewegung, ABBKartesisch;

procedure TForm3.Button1Click(Sender: TObject);
begin
   Form1.Close;
end;
      // Schließen-Button einstellen

procedure TForm3.Button2Click(Sender: TObject);
begin
   Form1.hContext.Connection.IOHandler.WriteLn
      ('movel(p[300,0,400,-180,0,180],v=30)');
   Timer1.Enabled:=True;
end;
      // Start-Button einstellen und Timer1 aktivieren

procedure TForm3.Button3Click(Sender: TObject);
begin
   Form1.hContext.Connection.IOHandler.WriteLn('exit');
end;
      // Beenden-Button einstellen

procedure TForm3.Button4Click(Sender: TObject);
begin
   Memo1.Lines.Clear;
end;
      // Clear-Button einstellen
```

```
procedure TForm3.Button5Click(Sender: TObject);
begin
   a:=IntToStr(pos('[',Form1.Antwort));
   b:=IntToStr(pos(']',Form1.Antwort));
   c:=copy(Form1.Antwort,StrToInt(a),StrToInt(b)-2);
   d:=IntToStr(pos(',',c));
   zwei:=copy(c,2,StrToInt(d)-2);
   e:=copy(c,StrToInt(d)+1,length(c)-StrToInt(d));
   f:=IntToStr(pos(',',e));
   eins:=copy(e,1,StrToInt(f)-1);
   g:=copy(e,StrToInt(f)+1,length(e)-StrToInt(f));
   h:=IntToStr(pos(',',g));
   drei:=copy(g,1,StrToInt(h)-1);
      // String-Antwort zerschneiden
      // Kommas suchen
      // Siehe Anlage, String-Antowort zerschneiden

   eins:=StringReplace (eins, '.', ',', []) ;
      // StringReplace in der Y-Position der Punkt durch das Komma erset-
      zen, um die StrToFloat-Funktion zu erkennen.

   if (0.5*1000*StrToFloat(Label2.text)<150) and
      (0.5*1000*StrToFloat(Label2.text)>-150) then
   eins2:=FloatToStr(0)
      // Wenn der Neigungswinkel auf der Y-Achse innerhalb von -27° bis
      27° liegt, ist er nicht geneigt, d.h. er ist immer noch horizontal
      und tut nichts.

   else if (0.5*1000*StrToFloat(Label2.text)>150) and
      (StrToFloat(eins)<250) then
   eins2:=FloatToStr(0.1*0.5*1000*StrToFloat(Label2.text))
      // Wenn der Neigungswinkel auf der Y-Achse größer als 27 ° ist und
      die Robo-ter_VerschiebungY weniger als 250mm beträgt, bewegt sich
      der Roboter gemäß dem Mo-tionSensordatum in Echtzeit.

   else if (0.5*1000*StrToFloat(Label2.text)>150) and
      (StrToFloat(eins)>250) then
   eins2:=FloatToStr(0)
      // Wenn der Neigungswinkel auf der Y-Achse größer als 27 ° ist und
      die Robo-ter_VerschiebungY größer als 250mm beträgt, bewegt sich
      der Robote in Y-Richtung nicht.

   else if (0.5*1000*StrToFloat(Label2.text)<-150) and
      (StrToFloat(eins)>-250) then
   eins2:=FloatToStr(0.1*0.5*1000*StrToFloat(Label2.text))
      // Wenn der Neigungswinkel auf der Y-Achse kleiner als -27 ° ist
      und die Roboter_VerschiebungY größer als -250mm beträgt, bewegt
      sich der Roboter gemäß dem MotionSensordatum in Echtzeit.

   else if (0.5*1000*StrToFloat(Label2.text)<-150) and
      (StrToFloat(eins)<-250) then
   eins2:=FloatToStr(0);
```

```
      // Wenn der Neigungswinkel auf der Y-Achse kleiner als -27 ° ist
      und die Robo-ter_VerschiebungY kleiner als -250mm beträgt, bewegt
      sich der Roboter in Y-Richtung nicht.

   zwei:=StringReplace (zwei, '.', ',', []) ;
   if (-0.5*1000*StrToFloat(Label3.text)<150) and
      (-0.5*1000*StrToFloat(Label3.text)>-150) then
   zwei2:=FloatToStr(0)
   else if (-0.5*1000*StrToFloat(Label3.text)>150) and
      (StrToFloat(zwei)<400) then
   zwei2:=FloatToStr(-0.1*0.5*1000*StrToFloat(Label3.text))
   else if (-0.5*1000*StrToFloat(Label3.text)>150) and
      (StrToFloat(zwei)>400) then
   zwei2:=FloatToStr(0)
   else if (-0.5*1000*StrToFloat(Label3.text)<-150) and
      (StrToFloat(zwei)>100) then
   zwei2:=FloatToStr(-0.1*0.5*1000*StrToFloat(Label3.text))
   else if (-0.5*1000*StrToFloat(Label3.text)<-150) and
      (StrToFloat(zwei)<100) then
   zwei2:=FloatToStr(0);

   if ((StrToFloat(eins)+0.1*0.5*1000*StrToFloat(Label2.text))>250) then
   eins2:=FloatToStrF(250-StrToFloat(eins), ffFixed, 8, 2)
   else if (StrToFloat(eins)+0.1*0.5*1000*StrToFloat(Label2.text))<-250
   then
   eins2:=FloatToStrF(-250-StrToFloat(eins), ffFixed, 8, 2);
      // Wenn die Y-Achse nach der Verschiebung den Grenzwert überschrei-
      tet, bewegt sich die Y-Achse schließlich nur noch zur Grenze.

   if (StrToFloat(zwei)-0.1*0.5*1000*StrToFloat(Label3.text))>400 then
   zwei2:=FloatToStrF(400-StrToFloat(zwei), ffFixed, 8, 2)
   else if (StrToFloat(zwei)-0.1*0.5*1000*StrToFloat(Label3.text))<100
   then
   zwei2:=FloatToStrF(100-StrToFloat(zwei), ffFixed, 8, 2);
      // Wenn die X-Achse nach der Verschiebung den Grenzwert überschrei-
      tet, bewegt sich die Y-Achse schließlich nur noch zur Grenze.

   if (StrToFloat(zwei2)<>0) or (StrToFloat(eins2)<>0) then
   begin
   eins2:=StringReplace (eins2, ',', '.', []);
   zwei2:=StringReplace (zwei2, ',', '.', []);
      Form1.hContext.Connection.IOHandler.WriteLn('movel(pose_trans(get_a
      ctual_tcp_pose(),['+zwei2+','+eins2+',0,0,0,0]),v=30)');
   end ;
      // Der Bewegungsbefehl wird gesendet, wenn mindestens eine Ver-
      schiebung in den zwei Achsenrichtungen vorliegt

   Memo1.Lines.Add('Nachricht: ' + Form1.Antwort);
   Memo1.Lines.Add('aktuell X: ' + zwei +' ---> Ziel X: +  ' + zwei2);
   Memo1.Lines.Add('aktuell Y: ' + eins +' ---> Ziel Y: +  ' + eins2);
end;

procedure TForm3.Button6Click(Sender: TObject);
```

```
begin
   Timer1.Enabled:=False;
   Label2.Text := '0';
   Label3.Text := '0';
end;
      // Stoppen-Button einstellen

procedure TForm3.Button7Click(Sender: TObject);
begin
   Timer1.Enabled:=True;
end;
      // Weiter-Button einstellen

procedure TForm3.Button8Click(Sender: TObject);
begin
      if (0.5*1000*StrToFloat(Label2.text)>150)
      or (0.5*1000*StrToFloat(Label2.text)<-150)
      or (-0.5*1000*StrToFloat(Label3.text)>150)
      or (-0.5*1000*StrToFloat(Label3.text)<-150)
      then
      begin
        if Runde=0 then
           Button5Click(self);
           Runde:=1;
      end;
      // Der Standardwert von Integer ' Runde ' ist 0. Wenn die Neigung
      der Tablet in vier Rich-tungen um mindestens eins größer ist als
      der Neigungsstartwert, wird Button5Click- Me-thoden aufgerufen und
      der Wert von Runde auf 1 geändert. Der Inhalt in Button5Click- Me-
      thoden ist konkrete Neigungseinstellungen.

        if Runde=1 then
        begin
           if   (-0.5*1000*StrToFloat(Label3.text)<150)
           and (-0.5*1000*StrToFloat(Label3.text)>-150)
           and (0.5*1000*StrToFloat(Label2.text)<150)
           and (0.5*1000*StrToFloat(Label2.text)>-150)
           then
           Runde:=0;
        end;
      // Nachdem der Wert von Runde auf 1 geändert wird, wenn das Tablet
      wieder horizontal wird, wird der Wert von Runde auf 0 zurückge-
      setzt.

end;

procedure TForm3.Button9Click(Sender: TObject);
begin
   Form1.hContext.Connection.IOHandler.WriteLn('movel(p[300,0,400,-
180,0,180],v=30)');
end;
      // Startposition-Button einstellen
```

```
procedure TForm3.TabItem1Click(Sender: TObject);
begin
   Form1.Show;
   Form1.TabItem1.IsSelected:=True;
end;
      // das Wechseln zwischen Form3 und Form1 realisieren

procedure TForm3.TabItem2Click(Sender: TObject);
begin
   Form2.Show;
   Form2.TabItem2.IsSelected:=True;
end;
      // das Wechseln zwischen Form3 und Form2 realisieren

procedure TForm3.TabItem3Click(Sender: TObject);
begin
   Form3.Show;
   Form3.TabItem3.IsSelected:=True;
end;
      // selbst anzeigen

procedure TForm3.Timer1Timer(Sender: TObject);
begin
   Label2.Text :=
      FloatToStrF(Motionsensor1.Sensor.AccelerationX,ffFixed, 8, 2);
   Label3.Text :=
      FloatToStrF(Motionsensor1.Sensor.AccelerationY,ffFixed, 8, 2);
   Button8Click(self);
end;
      // Die Zeitdauer von Timer1 wird auf 1 Millisekunden eingestellt,
      um dem Label Echtzeitwerte im OnTimer-Ereignis zuzuweisen und But-
      ton8Click-Methoden ständig aufzurufen. Der Inhalt in Button8Click-
      Methoden ist die Erstneigungseinstellung und Wiederneigungsein-
      stellung.

procedure TForm3.Timer2Timer(Sender: TObject);
begin
   Button5Click(self);
   Timer2.Enabled:=False;
end;
      // Die Zeitdauer von Timer2 wird auf 100 Millisekunden eingestellt,
      weil Button5Click-Methoden nur rechtzeitig aufgerufen werden kann.
      Der Inhalt in Button5Click-Methoden ist konkrete Neigungseinstel-
      lungen. Der Start von Timer2 ist abhängig von OnExecute-Ereignis
      des Form1.IdTCPServer1s. Der Ende von Timer2 ist abhängig von OnTi-
      mer-Ereignis des Timer2s.

      Auf diese Weise kann Button5Click-Methoden einmal aufgerufen wer-
      den, wenn der Ser-ver jedes Mal antwortet, dann wird der nächste
      Befehl gesendet usw.

end.
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 15. August 2018

Jiaqi Peng